

# Institutionen för systemteknik

## Department of Electrical Engineering

**Examensarbete**

### **Statistical Background Models with Shadow Detection for Video Based Tracking**

Examensarbete utfört i Datorseende  
vid Tekniska högskolan i Linköping  
av

**John Wood**

LITH-ISY-EX--07/3921--SE

Linköping 2007



**Linköpings universitet**  
**TEKNISKA HÖGSKOLAN**

Department of Electrical Engineering  
Linköpings universitet  
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola  
Linköpings universitet  
581 83 Linköping



# Statistical Background Models with Shadow Detection for Video Based Tracking

Examensarbete utfört i Datorseende  
vid Tekniska högskolan i Linköping  
av

**John Wood**


LITH-ISY-EX--07/3921--SE

Handledare: **Björn Johansson**  
isy, Linköpings universitet

Examinator: **Klas Nordberg**  
isy, Linköpings universitet

Linköping, 30 March, 2007



	<b>Avdelning, Institution</b> Division, Department  Computer Vision Laboratory Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	<b>Datum</b> Date  2007-03-30				
<b>Språk</b> Language  <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English  <input type="checkbox"/> _____	<b>Rapporttyp</b> Report category  <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	<b>ISBN</b> _____  <b>ISRN</b> LITH-ISY-EX--07/3921--SE  <b>Serietitel och serienummer ISSN</b> Title of series, numbering _____				
<b>URL för elektronisk version</b> <a href="http://www.cvl.isy.liu.se">http://www.cvl.isy.liu.se</a> <a href="http://www.ep.liu.se/2007/3921">http://www.ep.liu.se/2007/3921</a>						
<table border="0"> <tr> <td style="vertical-align: top;"><b>Titel</b> Title</td> <td>           Statistiska bakgrundsmodeller med detektion av skuggor för video baserad tracking             Statistical Background Models with Shadow Detection for Video Based Tracking         </td> </tr> <tr> <td style="vertical-align: top;"><b>Författare</b> Author</td> <td>           John Wood         </td> </tr> </table>			<b>Titel</b> Title	Statistiska bakgrundsmodeller med detektion av skuggor för video baserad tracking  Statistical Background Models with Shadow Detection for Video Based Tracking	<b>Författare</b> Author	John Wood
<b>Titel</b> Title	Statistiska bakgrundsmodeller med detektion av skuggor för video baserad tracking  Statistical Background Models with Shadow Detection for Video Based Tracking					
<b>Författare</b> Author	John Wood					
<b>Sammanfattning</b> Abstract  <p>A common problem when using background models to segment moving objects from video sequences is that objects cast shadow usually significantly differ from the background and therefore get detected as foreground. This causes several problems when extracting and labeling objects, such as object shape distortion and several objects merging together. The purpose of this thesis is to explore various possibilities to handle this problem.</p> <p>Three methods for statistical background modeling are reviewed. All methods work on a per pixel basis, the first is based on approximating the median, the next on using Gaussian mixture models, and the last one is based on channel representation. It is concluded that all methods detect cast shadows as foreground.</p> <p>A study of existing methods to handle cast shadows has been carried out in order to gain knowledge on the subject and get ideas. A common approach is to transform the RGB-color representation into a representation that separates color into intensity and chromatic components in order to determine whether or not newly sampled pixel-values are related to the background. The color spaces HSV, IHSL, CIELAB, YCbCr, and a color model proposed in the literature (Horprasert et al.) are discussed and compared for the purpose of shadow detection. It is concluded that Horprasert's color model is the most suitable for this purpose.</p> <p>The thesis ends with a proposal of a method to combine background modeling using Gaussian mixture models with shadow detection using Horprasert's color model. It is concluded that, while not perfect, such a combination can be very helpful in segmenting objects and detecting their cast shadow.</p>						
<b>Nyckelord</b> Keywords    Background Models, Gaussian Mixture Models, Shadow Detection, Color Spaces						



# Abstract

A common problem when using background models to segment moving objects from video sequences is that objects cast shadow usually significantly differ from the background and therefore get detected as foreground. This causes several problems when extracting and labeling objects, such as object shape distortion and several objects merging together. The purpose of this thesis is to explore various possibilities to handle this problem.

Three methods for statistical background modeling are reviewed. All methods work on a per pixel basis, the first is based on approximating the median, the next on using Gaussian mixture models, and the last one is based on channel representation. It is concluded that all methods detect cast shadows as foreground.

A study of existing methods to handle cast shadows has been carried out in order to gain knowledge on the subject and get ideas. A common approach is to transform the RGB-color representation into a representation that separates color into intensity and chromatic components in order to determine whether or not newly sampled pixel-values are related to the background. The color spaces HSV, IHSL, CIELAB, YCbCr, and a color model proposed in the literature (Horprasert et al.) are discussed and compared for the purpose of shadow detection. It is concluded that Horprasert's color model is the most suitable for this purpose.

The thesis ends with a proposal of a method to combine background modeling using Gaussian mixture models with shadow detection using Horprasert's color model. It is concluded that, while not perfect, such a combination can be very helpful in segmenting objects and detecting their cast shadow.





# Acknowledgments

This work has been carried out at the Computer Vision Laboratory at Linköping Institute of Technology. I would like to thank everybody working there for their hospitality and the friendly atmosphere.

I would especially like to thank my examiner Klas Nordberg and my supervisor Björn Johansson. Klas for introducing me to the field of computer vision and Björn for constantly discussing ideas and being extremely helpful and approachable.

I would also like to thank Autoliv Development for supporting this work.

*John Wood, March 2007.*



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Objectives . . . . .	2
1.3	Outline . . . . .	2
<b>2</b>	<b>Background Models</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Approximate Median Filtering . . . . .	6
	Algorithm . . . . .	6
	Discussion . . . . .	6
	Implementation . . . . .	7
2.3	Mixture Models . . . . .	9
2.3.1	Mixture Models . . . . .	9
2.3.2	Theoretical Derivation of Update Procedure . . . . .	11
	Expectation-Maximization Algorithm . . . . .	11
	Applying EM to Gaussian Mixture Models . . . . .	12
2.3.3	Stauffer and Grimson's Algorithm . . . . .	14
	Generating On-Line Averages . . . . .	14
	Practical Considerations . . . . .	16
	Discussion . . . . .	17
	Implementation . . . . .	18
2.4	Channel Representation . . . . .	21
	Algorithm . . . . .	22
	Discussion . . . . .	22
	Implementation . . . . .	22
<b>3</b>	<b>Shadow Detection</b>	<b>25</b>
3.1	Introduction . . . . .	25
3.2	Study of Existing Methods . . . . .	25
3.3	Color and Color Spaces . . . . .	26
3.3.1	Terminology . . . . .	26
3.3.2	Hue, Saturation, Value – HSV . . . . .	27
3.3.3	Improved Hue, Saturation, Lightness – IHSL . . . . .	28
3.3.4	Perceptually Uniform Color Space – CIELAB . . . . .	29

3.3.5	Color Space Used for Image and Video Encoding – YCbCr	30
3.3.6	Color Model of Horprasert et al. . . . .	31
3.4	Comparison . . . . .	31
3.4.1	Methods for Evaluation . . . . .	32
3.4.2	Test 1 . . . . .	33
3.4.3	Test 2 . . . . .	33
3.4.4	Test 3 . . . . .	34
3.5	Discussion . . . . .	35
<b>4</b>	<b>Proposed Method</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Proposed Method . . . . .	41
4.3	A Complete Implementation . . . . .	42
4.4	Results and Discussion . . . . .	45
4.4.1	A Three Hour Sequence . . . . .	45
4.4.2	A More Detailed Look . . . . .	49
4.4.3	Computational Performance . . . . .	51
4.5	Conclusions and Ideas for Future Work . . . . .	52
	<b>Bibliography</b>	<b>53</b>

# Chapter 1

## Introduction

In this chapter the background and motivation to the thesis are given. The objectives and the structure of the thesis are also outlined.

### 1.1 Motivation

This thesis concludes the master of science educational program in *Applied Physics and Electrical Engineering* at Linköping Institute of Technology.

The thesis is part of the research project *IVSS Intersection Accidents: Analysis and Prevention*. IVSS, or Intelligent Vehicle Safety Systems, is a program that is a joint venture by public sector agencies, private sector companies, and industry organizations set up to stimulate and further research and development regarding road safety for the future. The project Intersection Accidents: Analysis and Prevention aims to systematically study the chain of events which lead to accidents and/or near-accidents at intersections; partners in the project are Autoliv Development, Saab Automobile, Volvo Car Corporation, Linköpings University, and Chalmers Institute of Technology.

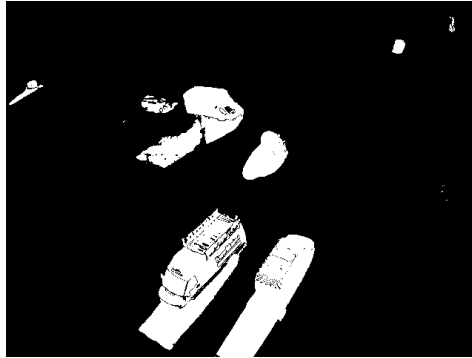
One part of the project aims to extract vehicle and traffic flow information from camera surveillance using image processing. The data collected will be used to test research hypotheses regarding driver behavior and environmental factors; as well as a basis for improving road safety at intersections. Some information about the project can be found at the projects website<sup>1</sup>.

One of the necessary steps in the current camera-based tracking is to estimate a binary image where the pixel-values represent background/foreground. The binary image is used to extract and label objects in the scene. Figure 1.1 shows an example of a binary background/foreground image.

A problem with the current method is that shadows often are detected as foreground. Problems caused by shadows being detected as foreground include object shape distortion and several objects merging together; therefore, it is desirable to develop a method that is aware of this problem and attempts to detect shadows.

---

<sup>1</sup><http://www.ivss.se/templates/ProjectPage.aspx?id=156>



**Figure 1.1.** Binary background/foreground image used to extract and label objects. White pixels correspond to object points and black pixels correspond to background points.

## 1.2 Objectives

The main objective of this thesis is to propose a method for background modeling that incorporates the detection and removal of shadows. Other objectives are to review and compare existing methods for both background modeling and shadow detection, as well as to implement selected algorithms as Matlab/C++ mex-files. The methods for background modeling that will be reviewed have been preselected whereas the methods for shadow detection will be selected by conducting a study of existing methods.

## 1.3 Outline

To accomplish the main objective the thesis has been outlined as follows. The thesis is divided into four chapters, introduction, background models, shadow detection, and proposed method.

**Chapter 1** provides an introduction to the thesis. The background and motivation to the thesis are given and the objectives are outlined.

**Chapter 2** introduces the concept of adaptive background models for video sequences and describes three methods for background modeling. The first is a simple method which approximates the median at each pixel. The second is a more complex, but also more powerful, method which utilizes Gaussian mixture models. A lot of focus is given on understanding the underlying theory of the method. The third and final method is based on channel representation and provides a simple algorithm comparable in performance to the second method. At the end of each section pseudo code of possible implementations as well as examples of typical results are given.

**Chapter 3** attempts to decide on a method for shadow detection by conducting a study of existing methods. It is decided that using a color model that separates color into luminance and chromaticity components will constitute the basis of the shadow detection algorithm. An overview of various color spaces, HSV, IHSL, CIELAB, YCrCb, as well as a color model proposed in the literature, is given. The chapter ends with some tests

and a discussion of the results.

**Chapter 4** proposes a solution to the main objective of the thesis, that is, a method for background modeling that incorporates shadow detection is proposed. The main focus is the final algorithm and its implementation. Pseudo code and an example of typical results with an accompanying discussion are given. The chapter ends with some conclusions and a summary of the ideas for future work that are mentioned in the thesis.





## Chapter 2

# Background Models

This chapter introduces the concept of adaptive background models for video sequences and describes three methods for background modeling. The first is a simple method which approximates the median at each pixel. The second is a more complex, but also more powerful, method which utilizes Gaussian mixture models. A lot of focus is given on understanding the underlying theory of the method. The third and final method is based on channel representation and provides a simple algorithm comparable in performance to the second method. At the end of each section pseudo code of possible implementations as well as examples of typical results are given.

### 2.1 Introduction

In computer vision a background model refers to an estimated image or the statistics of the background of a scene which an image or video sequence depicts. In object tracking from video sequences, i.e. tracking people, cars, etc., the background model plays a crucial role in separating the foreground from the background.

The simplest form of background model is perhaps taking an image of the scene when no objects are present and then using that image as the background model. The foreground can be determined by frame differencing, i.e. comparing each pixel in the currently sampled frame to the background image and if the difference is below some threshold, the pixel is classified as background. Such a solution may be sufficient in a controlled environment, but in an arbitrary environment such as outdoor scenes, light conditions will vary over time. Also, it may be either difficult or impossible to be able to take an image of the scene without any objects present. It is therefore highly desirable to have a background model that adapts to the scene regardless of its initial state.

This thesis focuses on adaptive background models that can be maintained in real-time. In some literature the adaptive methods explained here are referred to as recursive techniques, since the current background model is recursively updated in each iteration.

## 2.2 Approximate Median Filtering

A background model such as the one described above can further be categorized into what is called single mode background models. That is, it stores information regarding one mode of the background image's pixel values.

Approximate median filtering is a simple adaptive method for single mode background modeling. The method was developed due to the success of median filtering on a buffer of images, see [1] for more information, the method iteratively approximates the median and was originally described in [12].

### Algorithm

Let  $x_t$  denote the current sample value and let  $m_t$  denote the current approximation of the median,  $m_{t+1}$  is then approximated by

$$m_{t+1} = \begin{cases} m_t + \alpha & \text{if } x_t > m_t \\ m_t - \alpha & \text{if } x_t < m_t \\ m_t & \text{if } x_t = m_t, \end{cases} \quad (2.1)$$

where  $\alpha$  is set depending on the desired rate of convergence. For example, as the algorithm is described in [1],  $\alpha = 1$  is used; this is a high value. Consider grayscale pixel values in the range  $[0, 255]$  and let a pixel be classified as background if it falls within 15 units of the median. Furthermore, let an object move into that pixel. If that object is stationary for a minimum of 15 frames the background estimate is lost, which is how new stationary objects becomes part of the background image, but if the object eventually starts to move again the background image will need some time to return to its previous state. This can be seen as objects leaving trails in the background image. Setting  $\alpha$  depends on the situation, a high value of  $\alpha$  lets the background adapt to changes quickly, but if the scene has temporarily stationary objects, such as scenes of intersections, care needs be taken.

One way to arrive at (2.1) is to first consider how the median is normally calculated, the median for a data set  $\mathcal{X} = \{x_1, \dots, x_N\}$  can be determined by solving

$$\text{median}(\mathcal{X}) = \underset{m}{\operatorname{argmin}} \sum_{n=1}^N |x_n - m|. \quad (2.2)$$

The expression  $\sum_{n=1}^N |x_n - m| = e(m)$  can be thought of as an error function. Minimizing  $e(m)$  using gradient-based search gives the iterative solution

$$m_{t+1} = m_t - \alpha \frac{d}{dm_t} e(m_t) = m_t - \alpha \frac{d}{dm_t} \sum_{n=1}^N |x_n - m_t|. \quad (2.3)$$

Finally, writing (2.3) as an on-line cumulative average, one sample at a time, yields

$$m_{t+1} = m_t - \alpha \frac{d|x_t - m_t|}{dm_t} = \begin{cases} m_t + \alpha & \text{if } x_t > m_t \\ m_t - \alpha & \text{if } x_t < m_t \\ m_t & \text{if } x_t = m_t. \end{cases} \quad (2.4)$$

### Discussion

The main advantages of using this method is that it is very simple to implement, it handles speckle quite well since the median is used, and as a consequence of its simplicity it is also very efficient when implemented.

### Implementation

Using the notation in table 2.1 the core of the algorithm can be written in pseudo code as shown in algorithm 1.

An initial median estimate can be set in several ways, for example, the first frame of the video sequence can be used as an initial guess or it can be simply set to 0. To use the algorithm on multi-dimensional frames, e.g. color (RGB etc.), assuming each color channel, or dimension, is independent the median can be estimated separately for each dimension.

An example of typical result is given in example 2.1.

**Table 2.1.** Notation for algorithm 1.

$x_t$	pixel value at time t
$f_t$	frame at time t
$m$	median estimate
$\alpha$	adaptation rate
$T$	threshold
$\hat{B}$	binary background/foreground image

---

#### Algorithm 1 Approximate Median Filtering

---

```

1: # Approximate median.
2: for all  $x_t$  in  $f_t$  do
3:   if  $x_t > m$  then
4:      $m = m + \alpha$ 
5:   else if  $x_t < m$  then
6:      $m = m - \alpha$ 
7:   end if
8: end for
9:
10: # Segment binary background/foreground image.
11: for all  $x_t$  in  $f_t$  do
12:   if  $|x_t - m| \leq T$  then
13:      $\hat{B} = 1$ 
14:   else
15:      $\hat{B} = 0$ 
16:   end if
17: end for

```

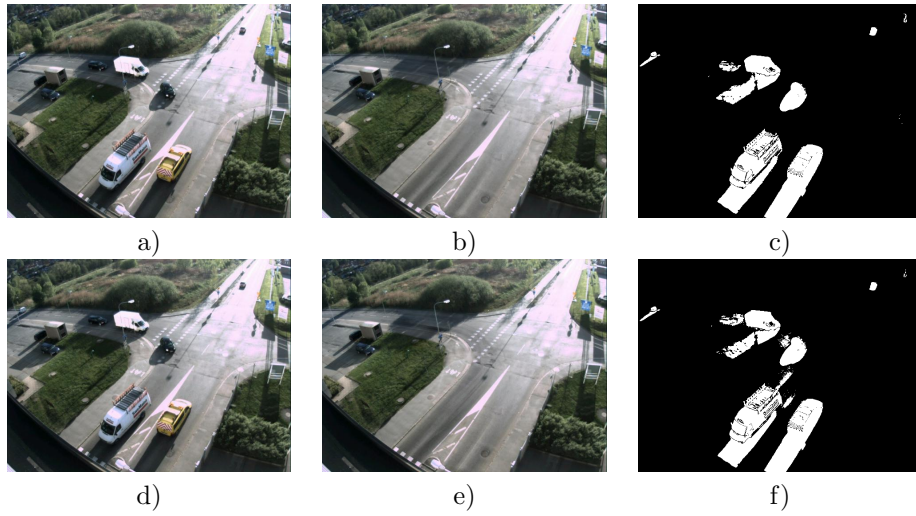
---

---

**Example 2.1**


---

In the following example, data from a typical intersection was used. The images were taken when the algorithm had performed 4450 iterations. The images in the top row of figure 2.1 shows the resulting estimated background image and the segmented binary foreground  $1 - \hat{B}$  using  $\alpha = 0.1$ , the images in the bottom row shows the result of setting  $\alpha = 1$ .



**Figure 2.1.** Result when using  $\alpha = 0.1$  (top row) and  $\alpha = 1$  (bottom row). a) and d) show the current frame, b) and e) show the current estimated background image, while c) and f) show the segmented foreground. Note the trails behind the white truck and the black car in f).

Note the trails behind the white truck and the black car when setting  $\alpha$  too high. Also note the shadows segmented as foreground, this will be discussed in the next chapter.

---

## 2.3 Mixture Models

The method for background modeling explored here was introduced by Stauffer and Grimson in [18]. Stauffer and Grimson noted that the computational performance of computers at the time, 1999, had reached a level where more complex and robust methods for real-time background modeling could be considered. Their approach is to use mixture models to represent the statistics of the scene. In contrast to approximate median filtering, using mixture models allows for a multi-modal background model which can be very useful in removing repetitive motion, e.g. shimmering water, leaves on a branch, or a swaying flag.

In the original paper the underlying mathematics are only briefly discussed. However, the method is quite flexible presuming an understanding of the mathematical theory, which is why a lot of attention is given this subject in the following sections.

### 2.3.1 Mixture Models

The pixel value measured by the camera sensor is the radiance<sup>1</sup> emitted from the surface point of first object to intersect that pixel's optical ray. In a dynamically changing scene (moving objects) the observed pixel value, therefore, depends on the surface of the possible intersecting objects as well as noise introduced by the camera.

In mathematical terms we can view the sampling process as the sampling of a random process  $\mathbf{X}$  where each sampled value is generated by the surface of some object. That is, the sampled value of  $\mathbf{X}$  is in fact an observation of a random variable  $\mathbf{S}_k$ , with  $k$  indicating which object was observed. Since only one of the objects can intersect the pixel's optical ray at a time except at object edges, the underlying events, that object  $k$  is observed, are disjoint. Therefore, the density function of  $\mathbf{X}$  can be modeled as linear combination of the density functions of the objects that generated the sample values. Such a model is called a mixture model and defines a mixture density as

$$p(x|\Theta) = \sum_{k=1}^K w_k p_k(x|\theta_k) , \quad (2.5)$$

where  $x$  is a sample of  $\mathbf{X}$  and  $\Theta$  is the parameter vector describing  $p$ , that is  $\Theta = \{w_1, \dots, w_K, \theta_1, \dots, \theta_K\}$ ,  $\theta_k$  in turn is a parameter vector describing  $p_k$ . Furthermore, the mixing densities  $w_k$  are in the range  $[0, 1]$  and  $\sum_{k=1}^K w_k = 1$ .  $p_k(x|\theta_k)$  are called component density functions and are normalized so that,

$$\int_{D_x} p_k(x|\Theta_k) dx = 1, \quad \text{for } k \in [1, \dots, K] . \quad (2.6)$$

These conditions ensure that the mixture density integrates to 1.

Example 2.2 shows a mixture model with defined parameter values.

---

<sup>1</sup>  $\frac{W}{stm^2}$  in SI units.

**Example 2.2**

Using 1-dimensional Gaussian component density functions with parameter vector  $\theta_k = \{\mu_k, \sigma_k\}$ , that is

$$p_k(x|\mu_k, \sigma_k) = \frac{1}{\sigma_k \sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}},$$

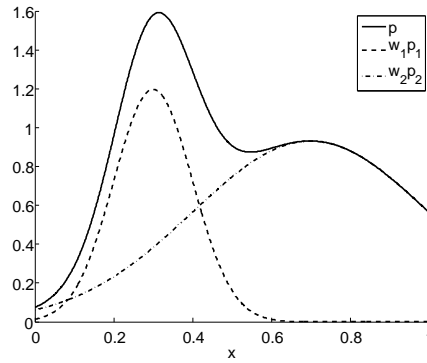
and setting

$$\begin{aligned} K &= 2, \\ w_k &= \{0.3, 0.7\}, \\ \mu_k &= \{0.3, 0.7\}, \\ \sigma_k &= \{0.1, 0.3\}, \end{aligned}$$

gives the mixture density

$$\begin{aligned} p(\mathbf{x}|\Theta) &= \sum_{k=1}^K w_k p_k(\mathbf{x}|\theta_k) \\ &= \frac{0.3}{0.1\sqrt{2\pi}} e^{-\frac{(x-0.3)^2}{2 \cdot 0.1^2}} + \frac{0.7}{0.3\sqrt{2\pi}} e^{-\frac{(x-0.7)^2}{2 \cdot 0.3^2}}. \end{aligned}$$

A graph of the mixture density where  $x$  has been restricted to the range  $[0, 1]$  is given in figure (2.2).



**Figure 2.2.** Mixture density.

From the graph we can see that the mixture density peaks at  $x \approx 0.35$ .

### 2.3.2 Theoretical Derivation of Update Procedure

Having decided on a representation of the probability density function of  $\mathbf{X}$ , the problem that remains is, given a data set of finite size of independently drawn samples of  $\mathbf{X}$ , how to choose parameter vector  $\Theta$ . A common approach to this problem is maximum likelihood estimation.

Recall the definition of the maximum likelihood estimation problem. Let  $\mathbf{X}$  denote a random variable with probability density function  $p(x|\Theta)$ ; furthermore, let  $\mathcal{X} = \{x_1, \dots, x_N\}$  denote a data set of  $N$  independently drawn samples of  $\mathbf{X}$ . The likelihood function is then

$$\mathcal{L}(\Theta|\mathcal{X}) = \prod_{i=1}^N p(x_i|\Theta) . \quad (2.7)$$

The optimal parameter vector is found by maximizing the likelihood function with respect to  $\Theta$ , i.e.

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \mathcal{L}(\Theta|\mathcal{X}) . \quad (2.8)$$

Solving (2.8) is often a difficult task, a commonly used trick is to maximize  $\log \mathcal{L}(\Theta|\mathcal{X})$  with respect to  $\Theta$  instead, since the product becomes a summation;  $\log \mathcal{L}(\Theta|\mathcal{X})$  is called the log-likelihood function. In the same spirit the EM algorithm can be used as a helpful aid when (2.8) is difficult to solve.

#### Expectation-Maximization Algorithm

The EM algorithm, or expectation-maximization algorithm, is a general iterative method for finding the maximum-likelihood estimate of the parameters of a given data set's governing distribution when that data set is incomplete. A data set is said to be incomplete when the governing distribution either has unknown, or hidden, parameters or missing values. The algorithm was introduced by Dempster et al. in [5].

Let  $\mathcal{X}$  be an incomplete data set that is known. By assuming the existence of a complete data set  $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$  a joint density function can be specified,

$$p(z|\Theta) = p(x, y|\Theta) = p(x|\Theta)p(y|x, \Theta) , \quad (2.9)$$

where  $\mathcal{Y}$  denotes the unknown data of  $\mathcal{Z}$ . Using this density function a new likelihood function, called the complete data likelihood function, can be defined as

$$\mathcal{L}(\Theta|\mathcal{Z}) = \mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = \prod_{i=1}^N p(x_i|\Theta)p(y_i|x_i, \Theta) . \quad (2.10)$$

By letting  $\Theta$  be fix, the complete data likelihood function becomes a function of the variable  $\mathcal{Y}$ , that is

$$\mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = /_{\Theta \text{ fix}} = l(\mathcal{Y}|\mathcal{X}, \Theta) . \quad (2.11)$$

Assuming  $\mathcal{Y}$  is an instance of a random variable  $\mathbf{Y}$ , we can calculate the expected value of  $\log l(\mathbf{Y}|\mathcal{X}, \Theta)$ . The EM algorithm uses a given set of parameters estimates  $\Theta_{t-1}$  to evaluate the marginal density of  $\mathbf{Y}$  given  $\mathcal{X}$  and uses this function to calculate the expected value of the complete data log-likelihood function; this step of the algorithm is known as the expectation step or E-step. Since  $\Theta$  is fix, the expected value will be a deterministic function of  $\Theta$ . Dempster et al. define a function  $Q$  as

$$Q(\Theta^t, \Theta^{t-1}) = E[\log l(\mathbf{Y}|\mathcal{X}, \Theta^t)|\mathcal{X}, \Theta^{t-1}] , \quad (2.12)$$

where  $\Theta^t = \Theta$  and the right hand side is evaluated by

$$E[\log l(\mathbf{Y}|\mathcal{X}, \Theta^t)|\mathcal{X}, \Theta^{t-1}] = \int_{\mathbf{y} \in D_{\mathbf{Y}}} \log l(\mathbf{y}|\mathcal{X}, \Theta^t) p(\mathbf{y}|\mathcal{X}, \Theta^{t-1}) d\mathbf{y} . \quad (2.13)$$

The second, and final, step of the EM algorithm is to maximize  $Q$  with respect to  $\Theta^t$ , that is

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} Q(\Theta^t, \Theta^{t-1}) . \quad (2.14)$$

This step of the algorithm is known as the maximization step or M-step.

The EM algorithm then uses  $\Theta^*$  as  $\Theta^{t-1}$  in (2.12) and the two steps of the algorithm are repeated. Each iteration of the algorithm is guaranteed to increase the likelihood function until a local maxima is reached. A mathematical proof of this statement as well as information regarding rate-of-convergence can be found in [5].

### Applying EM to Gaussian Mixture Models

A common approach to the problem of determining maximum likelihood parameter estimates for mixture models is to use the EM algorithm. Consider a random variable  $\mathbf{X}$  represented by a mixture model consisting of  $K$  underlying random processes each with their own probability density function, the mixture density is then

$$p(x|\Theta) = \sum_{k=1}^K w_k p_k(x|\theta_k) . \quad (2.15)$$

Again let  $\mathcal{X} = \{x_1, \dots, x_N\}$  denote a data set of  $N$  independently drawn samples of  $\mathbf{X}$ . Since the samples are independently drawn and identically distributed by  $p(x|\Theta)$ , the log-likelihood function becomes

$$\log \mathcal{L}(\Theta|\mathcal{X}) = \log \prod_{i=1}^N p(x_i|\Theta) = \sum_{i=1}^N \log p(x_i|\Theta) = \sum_{i=1}^N \log \sum_{k=1}^K w_k p_k(x_i|\theta_k) . \quad (2.16)$$

Equation (2.16) is difficult to maximize. However, assuming the underlying processes are disjoint, (2.16) can be greatly simplified if it is known which process generated each sample; let  $k_i \in \{1, \dots, K\}$  be a variable that is known, indicating this information. I.e. if  $k_i = k$  the  $i$ :th sample was generated by the  $k$ :th process. The expression then reduces to

$$\log \mathcal{L}(\Theta|\mathcal{X}) = \sum_{i=1}^N \log w_{k_i} p_{k_i}(x_i|\theta_{k_i}) . \quad (2.17)$$

Therefore, assume that the data set  $\mathcal{X}$  is incomplete in the sense that an accompanying unobserved data set indicating which process generated each sample value exists. Call this missing data set  $\mathcal{K} = \{k_1, \dots, k_N\}$ , and let  $\mathcal{Z} = (\mathcal{X}, \mathcal{K})$  denote the complete data set. If  $\mathcal{K}$  is known the complete data log-likelihood becomes



$$\log \mathcal{L}(\Theta|\mathcal{X}, \mathcal{K}) = \sum_{i=1}^N \log w_{k_i} p_{k_i}(x_i|\theta_{k_i}) . \quad (2.18)$$

However,  $\mathcal{K}$  is unknown, but if we, accordingly with the EM algorithm, assume that  $\mathcal{K}$  is an instance of a random variable  $\mathbf{K}$ , we can proceed with the expectation step of the algorithm. All that is needed is an expression for the marginal density of  $\mathbf{K}$  given  $\mathcal{X}$ . Note that in this case, a sample  $\mathbf{k}$  of  $\mathbf{K}$  will be a vector of  $N$  elements. Using Bayes' rule it is easy to find an expression for the conditional probability of  $k_i$  given  $x_i$  and a parameter vector  $\Theta^{t-1}$

$$p(k_i|x_i, \Theta^{t-1}) = \frac{w_{k_i} p_{k_i}(x_i|\theta_{k_i}^{t-1})}{\sum_{k=1}^K w_k p_k(x_i|\theta_k^{t-1})} . \quad (2.19)$$

The mixing parameters  $w_k$  can in this case be thought of as prior probabilities  $p(k_i|\theta^{t-1})$ . Since the samples are independently drawn, the marginal density is

$$p(\mathcal{K}|\mathcal{X}, \Theta^{t-1}) = \prod_{i=1}^N p(k_i|x_i, \Theta^{t-1}) . \quad (2.20)$$

Equation (2.12) takes the form

$$\begin{aligned} Q(\Theta^t, \Theta^{t-1}) &= E[\log l(\mathbf{K}|\mathcal{X}, \Theta^t)|\mathcal{X}, \Theta^{t-1}] \\ &= \sum_{\mathbf{k} \in D_{\mathbf{K}}} \log l(\mathbf{k}|\mathcal{X}, \Theta^t) p(\mathbf{k}|\mathcal{X}, \Theta^{t-1}) \\ &= \text{quite a bit of algebra...} \\ &= \sum_{k=1}^K \sum_{i=1}^N \log(w_k p_k(x_i|\theta_k^t)) p(k|x_i, \Theta^{t-1}) . \end{aligned} \quad (2.21)$$

We have now analytically performed the E-step.

With the exception of determining  $w_k$ , performing the M-step is a matter of choosing component density functions and differentiating  $Q(\Theta^t, \Theta^{t-1})$  with respect to the various parameters and setting them to 0. It is not necessary to specify component density functions in order to determine  $w_k$ . Blimes shows in [2] that when using Gaussian component density functions, it is possible to find analytic expressions for all of the new parameters  $\Theta^t$  as functions of the old parameters  $\Theta^{t-1}$ . Using d-dimensional Gaussian with parameter vectors  $\theta_k = \{\mu_k, \Sigma_k\}$ , i.e.,

$$p_k(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)} , \quad (2.22)$$

gives the expressions

$$w_k^t = \frac{1}{N} \sum_{i=1}^N p(k|x_i, \Theta^{t-1}) , \quad (2.23)$$

$$\mu_k^t = \frac{\sum_{i=1}^N x_i p(k|x_i, \Theta^{t-1})}{\sum_{i=1}^N p(k|x_i, \Theta^{t-1})} , \quad (2.24)$$

$$\Sigma_k^t = \frac{\sum_{i=1}^N p(k|x_i, \Theta^{t-1})(x_i - \mu_k^t)(x_i - \mu_k^t)^T}{\sum_{i=1}^N p(k|x_i, \Theta^{t-1})} . \quad (2.25)$$

Note the use of the new parameter estimate  $\mu_k^t$  in the expression for  $\Sigma_k^t$ .

An alternate approach to derive (2.23)-(2.25) can be found in [3], where Gaussian component density functions and a diagonal covariance matrix are used from an early stage in the derivation, and convergence for this special case is shown.

### 2.3.3 Stauffer and Grimsons Algorithm

Now that we have the mathematical understanding of how the EM algorithm is applied to Gaussian mixture models we can move on to look at the algorithm Stauffer and Grimson presented in [18].

#### Generating On-Line Averages

Since we expect the scene to dynamically change over time, the pixel process  $\mathbf{X}$  can not be considered a stationary process over a longer period of time; however, by specifying  $N$  only a finite history of samples will be used to determine the parameter vector  $\Theta$ . Equations (2.23)-(2.25) needs to be evaluated for every pixel in every video frame. Since this is a computationally cumbersome operation when  $N$  is big, a trick that simplifies the operation is to compute the averages recursively.

By evaluating the expressions to  $N + 1$  the  $(N + 1) : th$  term can be extracted. The mixing densities in (2.23) evaluated to  $N + 1$  are

$$\begin{aligned} w_k^{N+1} &= \frac{1}{N+1} \sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1}) \\ &= \frac{1}{N+1} \sum_{i=1}^N p(k|x_i, \Theta^{t-1}) + \frac{1}{N+1} p(k|x_{N+1}, \Theta^{t-1}) \\ &= \frac{N}{N+1} w_k^N + \frac{1}{N+1} p(k|x_{N+1}, \Theta^{t-1}) \\ &= (1 - \frac{1}{N+1}) w_k^N + \frac{1}{N+1} p(k|x_{N+1}, \Theta^{t-1}) \\ &= (1 - \alpha) w_k^N + \alpha \cdot p(k|x_{N+1}, \Theta^{t-1}) , \end{aligned} \quad (2.26)$$

where  $\alpha = \frac{1}{N+1}$ . Applying the same technique to (2.24) yields

$$\begin{aligned}
\mu_k^{N+1} &= \frac{\sum_{i=1}^{N+1} x_i p(k|x_i, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} \\
&= \frac{\sum_{i=1}^N x_i p(k|x_i, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} + \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} x_{N+1} \\
&= \frac{\sum_{i=1}^N p(k|x_i, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} \mu_k^N + \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} x_{N+1} \\
&= \frac{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1}) - p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} \mu_k^N + \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} x_{N+1} \\
&= \left(1 - \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})}\right) \mu_k^N + \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} x_{N+1} \\
&= (1 - \rho_k) \mu_k^N + \rho_k x_{N+1} ,
\end{aligned} \tag{2.27}$$

where

$$\rho_k = \frac{p(k|x_{N+1}, \Theta^{t-1})}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} = \frac{p(k|x_{N+1}, \Theta^{t-1})}{(N+1) \cdot w_k^{N+1}} = \frac{\alpha \cdot p(k|x_{N+1}, \Theta^{t-1})}{w_k^{N+1}} . \tag{2.28}$$

Finally, applying the same algebraic tricks to (2.25) yields

$$\begin{aligned}
\Sigma_k^{N+1} &= \frac{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1}) (x_i - \mu_k^{N+1}) (x_i - \mu_k^{N+1})^T}{\sum_{i=1}^{N+1} p(k|x_i, \Theta^{t-1})} \\
&= \dots \\
&= (1 - \rho_k) \Sigma_k^N + \rho_k (x_{N+1} - \mu_k^{N+1}) (x_{N+1} - \mu_k^{N+1})^T .
\end{aligned} \tag{2.29}$$

So, to summarize, writing (2.26)-(2.29) as on-line cumulative averages in terms of new and old parameters we have

$$w_k^{new} = (1 - \alpha) w_k + \alpha \cdot p(k|x_t, \Theta^{old}) , \tag{2.30}$$

$$\mu_k^{new} = (1 - \rho_k) \mu_k + \rho_k x_t , \tag{2.31}$$

$$\Sigma_k^{new} = (1 - \rho_k) \Sigma_k + \rho_k (x_t - \mu_k^{new}) (x_t - \mu_k^{new})^T , \tag{2.32}$$

where

$$\alpha = \frac{1}{N+1} , \tag{2.33}$$

$$\rho_k = \frac{\alpha \cdot p(k|x_t, \Theta^{old})}{w_k^{new}} . \tag{2.34}$$

When setting  $\alpha$  to a constant value these equations approximately implement (2.23)-(2.25) and are very useful since using (2.30) instead of (2.23) allows us to compute the new mixing densities without storing the previous sample values. These equations differ somewhat from the equations Stauffer and Grimson presented in their paper. Most notably, (2.34) is different, this is however the same result as in [13].

### Practical Considerations

The first step of the algorithm is to estimate which of the  $K$  underlying distributions generated the current pixel value  $x_t$  and update its parameters. Stauffer and Grimson define a match as pixel value falling within  $\lambda = 2.5$  standard deviations of a distribution, which can be interpreted in terms of Mahalanobis distance as

$$d_k^2 = (x_t - \mu_k)^T \Sigma_k^{-1} (x_t - \mu_k) , \quad (2.35)$$

$$M_k = \begin{cases} 1 & \text{if } d_k < \lambda \\ 0 & \text{otherwise.} \end{cases} \quad (2.36)$$

When using  $\lambda = 2.5$  and 1-dimensional Gaussians (e.g. gray scale images) this value accounts for 98.76% of the Gaussians possible values. Equation (2.36) is essentially an approximation of  $p(k|x_t, \Theta^{old})$ , noting that in practise,  $p(k|x_t, \Theta^{old})$  is approximately 1 for distributions with a mean close to the generated pixel value and approximately 0 otherwise.

Note that Stauffer and Grimson use a single scalar to represent their covariance matrix, that is,  $\Sigma_k = \sigma_k^2 \mathbf{I}$ , where  $\mathbf{I}$  is the identity matrix. This is done to simplify the matrix inversion in (2.35) since it can easily be performed analytically using this simplification. Also note that the inversion can be easily performed analytically if  $\Sigma_k$  is assumed to be diagonal, which is an assumption that holds true if the dimensions are independent, and therefore, one that is often reasonable. For the remainder of this section a full covariance matrix will still be used, so the reader can apply whatever assumption suits his or her needs.

If several distributions match, the highest peaking distribution is chosen, this is done by choosing the  $k$  which maximizes  $w_k / \|\Sigma_k\|_F$ , where  $F$  denotes the Frobenius norm<sup>2</sup>, which will be a high value for distributions that often generate pixel values, and thus, are likely to have generated the current value. Its parameters are then updated using (2.30)-(2.32), and the mixture densities are renormalized so that they sum up to one.

The only computationally bothersome operation when calculating the new parameters is the evaluation of  $\rho_k$ . Using the approximation that  $p(k|x_t, \Theta^{old})$  is approximately 1 if the distribution is a match and 0 otherwise yields

$$\rho_k \approx \frac{\alpha}{w_k^{new}} . \quad (2.37)$$

Another approximation that can be made which is even simpler than (2.37) is to use (2.30) in (2.34), that is,

$$\begin{aligned} \rho_k &= \frac{\alpha \cdot p(k|x_{N+1}, \Theta^{old})}{w_k^{N+1}} \\ &= \frac{w_k^{new} - (1 - \alpha)w_k}{w_k^{new}} \\ &= 1 - (1 - \alpha) \frac{w_k}{w_k^{new}} \approx \alpha , \end{aligned} \quad (2.38)$$

where  $w_k / w_k^{new} \approx 1$ , which is true as long as  $w_k \gg \alpha$ . Both approximation (2.37) and (2.38) are discussed in [13]. A problem with the first approximation is that  $\rho_k = 1$  if  $w_k^{new} = \alpha$  which gives a covariance matrix that is  $\mathbf{0}$ , where  $\mathbf{0}$  is the zero matrix. The

---

<sup>2</sup>Defined as  $\|A\|_F^2 = \sum_{i,j} a_{ij}^2$

second approximation does not have this problem. However, distributions can rarely match, and the approximation that  $w_k \gg \alpha$  may not hold since  $w_k^{new} \approx \alpha$  if  $w_k \approx 0$ , and therefore,  $p_k \approx \alpha$  is invalid. The first approximation is safest to use, but care may need to be taken if  $\rho_k \approx 1$ .

If no distributions can be considered a match, the lowest peaking distribution is replaced by a new distribution, using the current pixel value as its mean, a high covariance matrix, and a low mixing density. This is how new objects become part of the background.

Stauffer and Grimson maintain the distributions ordered with respect to  $\frac{w_k}{\sigma_k}$ , using a full covariance matrix this expression becomes  $\frac{w_k}{\|\Sigma\|_F}$ . This is useful when determining which distributions constitute background. The first  $B$  distributions to have a combined mixing density greater than a threshold  $T$  are chosen as the background model, that is,

$$B = \underset{b}{\operatorname{argmin}} \left( \sum_{k=1}^b w_k > T \right) . \quad (2.39)$$

The binary foreground image is then estimated by calculating the Mahalanobian distance for the  $B$  first distribution. If the pixel can't be considered a match, in the same sense as above, it is deemed foreground.

## Discussion

One of the main advantages of using this method is the possibility of handling multi-modal background models. The mathematical derivation of the update procedure can be intimidating without having previous experience with the EM algorithm and/or mixture models. However, the resulting update procedure is fairly simple and intuitive. As a result, implementing the algorithm is moderately complicated and it performs quite well.

A problem that occurred when implementing the algorithm was that the mixture model of a pixel can get caught in a vicious circle due to the renormalization of the mixing densities. Since the lowest peaking distribution is replaced by a new distribution with a high covariance matrix and a low mixing density, the remaining densities will get an undeserving boost if the replaced distribution has a higher mixing density than the new distribution, i.e.  $w_k > w_{init}$ . One solution to this problem is to not renormalize the mixing densities when a distribution is replaced and setting its initial density to  $\alpha$ . This way the mixing densities sum up to at most 1 and never-matching distributions are allowed to fade out.

Another practical observation is that the covariance vector will adapt to the noise of the equipment, which can be very small, causing the method to be very sensitive to small changes. This can easily be avoided by setting a lower limit on the terms of the covariance vector, which has a regularizing effect.

Extending the algorithm to use a diagonal covariance matrix instead of a single variance opens up some new possibilities. Since using a diagonal covariance is equivalent to assuming independent dimensions, we can combine different features in the same input vector and use the algorithm without alteration. For instance, some interesting features could be thermal information from an IR-camera, optical flow, or depth (stereo-vision). Another possibility is that we do not necessarily have to use pixels as input, we could use regions as input instead, e.g. for each pixel reshape a 5x5 region centered at the current pixel to a 25-element vector and use that as input.

It is important to note that the parameter  $\lambda$  has different meaning depending on the dimension  $d$  of the input data. As previously mentioned, in 1D  $\lambda = 2.5$  captures 98.76% of the values originating from a Gaussian distribution, this value can be determined by evaluating  $e_1(\lambda) = \int_{-\lambda\sigma}^{\lambda\sigma} p(x|\mu, \sigma) dx$  using numerical methods. Recall that the allowed

values of  $x$  are defined by values falling within  $\lambda$  standard deviations of the mean. In 2D, an analytic solution can be found using polar coordinates. The solution is  $e_2(\lambda) = 1 - e^{-\lambda^2/2}$  and  $\lambda = 2.5$  gives the value 95.61%. In general it is difficult to evaluate  $e_d(\lambda)$ ; it can be said, though, that the value decreases with increasing  $d$ .<sup>3</sup> When using high dimensional input data it may be a good idea to use a different distance metric.

### Implementation

Algorithm 2 and 3 show pseudo code of a possible implementation using a diagonal covariance and approximation (2.37), that is  $p(k|x_t, \Theta^{old}) \approx 1$  if the  $k$ :th Gaussian matches and zero otherwise. The notation used is explained in table 2.2 and an example of some typical results is given in example 2.3.

**Table 2.2.** Notation for algorithm 2 and 3.

$x_t$	pixel value at time $t$
$f_t$	frame at time $t$
$w$	mixing density
$\mu$	mean
$\sigma^2$	variance
$K$	number of mixture components
$D$	dimension of mixture components
$\alpha$	learning rate
$\lambda$	threshold
$w_{init}$	initial mixing density
$\sigma_{init}^2$	initial covariance vector
$T$	threshold
$k$	integer in the range $[1, \dots, K]$
$m$	integer in the range $[1, \dots, K]$
$match$	boolean
$B$	background model
$\hat{B}$	binary background/foreground image
$\circ$	element-wise multiplication

---

<sup>3</sup>Assuming a diagonal covariance matrix  $e_d(\lambda) \leq \int_{-\lambda\sigma_1}^{-\lambda\sigma_1} \dots \int_{-\lambda\sigma_d}^{-\lambda\sigma_d} p(x|\mu, \Sigma) dx_1 \dots dx_d$ , gives an upper estimate, with equality only for  $d = 1$ .

---

**Algorithm 2** Background modeling using a mixture of D-dimensional Gaussians

---

```

1: # Update Model
2: for all  $x_t$  in  $f_t$  do
3:   for all  $k \in [1, \dots, K]$  do
4:      $d_k^2 = \sum_{d=1}^D \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$ 
5:     if  $d_k < \lambda$  then
6:       if  $match = 0$  then
7:          $m = k$ 
8:       else if  $\frac{w_k}{\sqrt{\|\sigma_k^2\|}} > \frac{w_m}{\sqrt{\|\sigma_m^2\|}}$  then
9:          $m = k$ 
10:      end if
11:       $match = 1$ 
12:    end if
13:  end for
14:
15:  if  $match = 0$  then
16:     $m = K$ 
17:     $w_m = w_{init}$ 
18:     $\mu_m = x_t$ 
19:     $\sigma_m^2 = \sigma_{init}^2$ 
20:  else
21:     $w_m = (1 - \alpha)w_m + \alpha$ 
22:     $\rho_m = \frac{\alpha}{w_m}$ 
23:     $\mu_m = (1 - \rho_m)\mu_m + \rho_m \cdot x_t$ 
24:     $\sigma_m^2 = (1 - \rho_m)\sigma_m^2 + \rho_m(x_t - \mu_m) \circ (x_t - \mu_m)$ 
25:  end if
26:
27:  Renormalize  $(w_1, \dots, w_K)$  to sum up to 1.
28:
29:  if  $match \neq 0$  then
30:    Sort  $w, \mu, \sigma$  with respect to  $(\frac{w_1}{\sqrt{\|\sigma_1^2\|}}, \dots, \frac{w_K}{\sqrt{\|\sigma_K^2\|}})$ .
31:  end if
32:
33:   $B = \operatorname{argmin}_b \left( \sum_{k=1}^b w_k > T \right)$ 
34: end for

```

---

---

**Algorithm 3** Background segmentation for algorithm 2.

---

```

1: for all  $x_t$  in  $f_t$  do
2:    $\hat{B} = 0$ 
3:   for all  $k \in [1, \dots, B]$  do
4:      $d_k^2 = \sum_{d=1}^D \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$ 
5:     if  $d_k < \lambda$  then
6:        $\hat{B} = 1$ 
7:     end if
8:   end for
9: end for

```

---

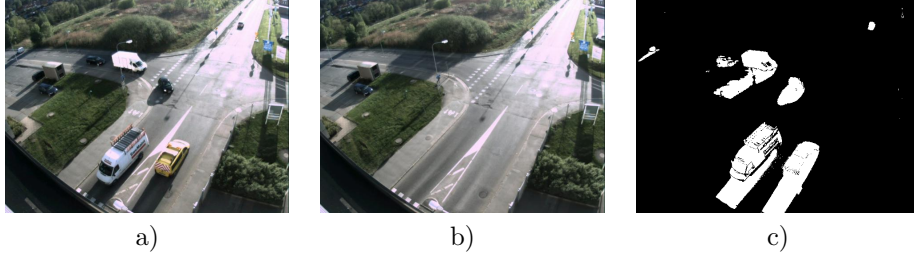


---

**Example 2.3**

---

The following example shows some typical output from the algorithm using the same data as in example 2.1.



**Figure 2.3.** Result when using  $K = 4$ ,  $\alpha = \frac{1}{600}$ , and  $T = 0.8$ . a) current frame, b) most probable background image, c) segmented foreground.

Shadows are still a problem. There is a swaying flag in the upper right corner, inspection of the data reveals that several of the pixels in that area have been recognized as bi-modal.

---



## 2.4 Channel Representation

As an alternative to Stauffer and Grimsons algorithm, a channel representation can be used to model the statistics of a scene. An overview of channel representation is given in [8]. For this section it is sufficient to think of channel representation as a variant of (2.5) where  $p_k(x|\theta_k)$  is called a channel function and has a fixed position and width; therefore, there is no parameter vector  $\theta_k$  in this case and a channel function will be referred to as only  $p_k(x)$ . An example of a channel function is

$$p_k(x) = \begin{cases} \cos^2(\frac{|x-c_k|}{\omega}\pi) & \text{if } \frac{|x-c_k|}{\omega} < 1/2 \\ 0 & \text{otherwise.} \end{cases} \quad (2.40)$$

where  $\omega$  is the channel width and  $c_k$  is the channel center. The main parameter vector in (2.5) is in this case  $\Theta = \{w_1, \dots, w_k\}$ . Gaussians and splines have also been used in different applications.

The channel representation of a value  $x$  is denoted by a channel vector  $u$  where each element is the value of the channel function for that value of  $x$ . The process of determining the vector  $u$  is called channel encoding. For example, using the channel functions mentioned above, and setting the channel centers  $c_k = \{1, \dots, 10\}$  and  $\omega = 3$ , the channel representation of  $x = 3.3$  is

$$u(3.3) = (0 \quad 0.043 \quad 0.905 \quad 0.552 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T, \quad (2.41)$$

where each element is evaluated by  $u_k = p_k(x)$ . The vector can be visualized as in figure 2.4, where the dashed curves are the channel functions  $p_k(x)$  and the solid curve is the sum of the channel functions weighed by the elements of  $u(x)$ , that is  $\sum_{k=1}^K u_k p_k(x)$ . The inverse process is called channel decoding and is a bit more complicated, see [8] for more information.

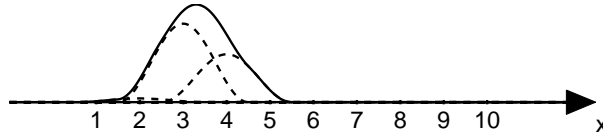


Figure 2.4.  $u(3.3)$ .

A very useful property of channel representations is that a channel vector can represent several values simultaneously as long as the distances between the values are sufficiently large. For instance, using the same parameters as above, the sum of the channel representation of the values  $x_1 = 3.3$  and  $x_2 = 8.1$  is

$$u = (0 \quad 0.043 \quad 0.905 \quad 0.552 \quad 0 \quad 0 \quad 0.1654 \quad 0.9891 \quad 0.3455 \quad 0)^T, \quad (2.42)$$

and can be visualized as in figure 2.5.

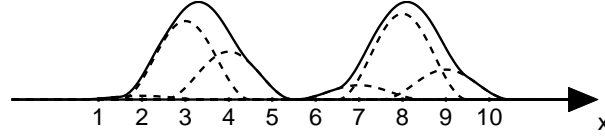


Figure 2.5.  $u(3.3) + u(8.1)$ .

### Algorithm

A background model maintained by, at each pixel in the video sequence, averaging the channel representation of the sample values for that pixel. The channel representation of a data set  $\mathcal{X} = \{x_1, \dots, x_N\}$  of  $N$  sample values is recursively calculated as

$$w^{new} = (1 - \alpha)w + \alpha \cdot p, \quad (2.43)$$

where  $\alpha$  is the learning rate and  $p$  is the channel representation of the current sample which is calculated as

$$p(x) = (p_1 \ p_2 \ \dots \ p_K)^T, \quad (2.44)$$

where  $p_k = p_k(x)$ .

The binary background/foreground image is then generated by calculating the dot product of the normalized vectors  $\hat{w}$  and  $\hat{p}$ , that is,

$$c = \hat{w}^T \hat{p}. \quad (2.45)$$

The dot product will be 1 if the vectors are equal and 0 if they are orthogonal. If  $c$  is higher than some threshold  $T$  the pixel is classified as background.

### Discussion

As can be seen in the previous section, channel representation is a very useful tool to represent a multi-modal background model, an algorithm comparable to Stauffer and Grimson can be expressed in only a few short steps.

In contrast to using mixture models, the number of channel functions needed to achieve good discrimination when dealing with several modes is high, and consequently, the algorithm uses a lot of memory. Depending on the size of the frames in the video used this can be a serious disadvantage. However, the channel vectors often contain many elements that are 0, and utilizing a sparse data format may remedy this problem. Also, using a fixed width  $\omega$  gives a less adaptive model.

### Implementation

Algorithm 4 and 5 show pseudo code of a possible implementation using a diagonal covariance and approximation (2.37), the notation used is explained in table 2.3. Typical output from the algorithm can be seen in example 2.4.

**Table 2.3.** Notation for algorithms 4 and 5.

$x_t$	pixel value at time $t$
$f_t$	frame at time $t$
$K$	number of channels
$\omega$	channel width, fixed and common for all channels
$c_k$	channel centers, fixed and evenly spread in the range of $x$ (e.g. $\{c_k\} = \{x_{min}, x_{min} + \omega, x_{min} + 2\omega, \dots, x_{max}\}$ )
$\alpha$	learning rate
$T$	threshold
$w$	background model
$p$	channel representation of $x_t$
$\hat{B}$	binary background/foreground image

**Algorithm 4** Background modeling using a set of channels.

---

```

1: for all  $x_t$  in  $f_t$  do
2:   # Channel encode  $x_t$ 
3:   for all  $k \in [1, \dots, K]$  do
4:      $d_k = \frac{|x_t - c_k|}{\omega}$ 
5:     if  $d_k < 1/2$  then
6:        $p_k = \cos^2(d_k \pi)$ 
7:     else
8:        $p_k = 0$ 
9:     end if
10:  end for

11:  # Update background model
12:  for all  $k \in [1, \dots, K]$  do
13:     $w_k = (1 - \alpha)w_k + \alpha p_k$ 
14:  end for
15: end for

```

---

**Algorithm 5** Background segmentation for algorithm 4

---

```

1: for all  $x_t$  in  $f_t$  do
2:   # Channel encode  $x_t$  (already available in algorithm 4)
3:   for all  $k \in [1, \dots, K]$  do
4:      $d_k = \frac{|x_t - c_k|}{\omega}$ 
5:     if  $d_k < 1/2$  then
6:        $p_k = \cos^2(d_k \pi)$ 
7:     else
8:        $p_k = 0$ 
9:     end if
10:  end for

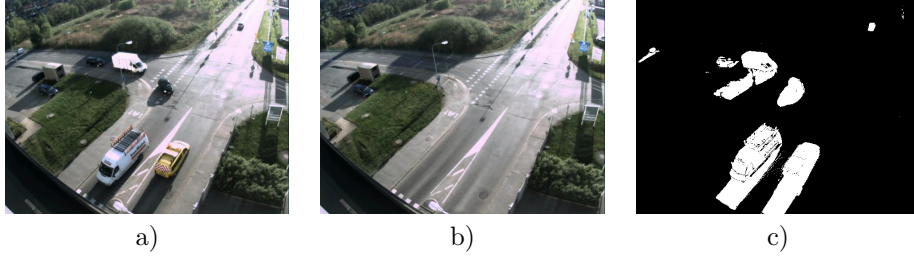
11:  # Compare to background model
12:   $corr = \left( \sum_{k=1}^K w_k p_k \right) / \sqrt{\sum_{k=1}^K w_k^2} / \sqrt{\sum_{k=1}^K p_k^2}$ 
13:  if  $corr \geq T$  then
14:     $\hat{B} = 1$ 
15:  else
16:     $\hat{B} = 0$ 
17:  end if
18: end for

```

---

**Example 2.4**

The following example shows some typical output from the algorithm using the same data as in example 2.1 and 2.3.



**Figure 2.6.** Result when using  $K = 11$ ,  $\omega = 0.2$ ,  $\alpha = \frac{1}{200}$ , and  $T = 0.5$ . a) current frame, b) most likely background image, c) segmented foreground.

A result very similar to that of example 2.3; the foreground looks a bit more homogeneous.

---

## Chapter 3

# Shadow Detection

In this chapter various methods for shadow detection are evaluated and one method is selected to be implemented together with one of the background modeling algorithms discussed in the previous chapter as a solution to the main objective. The chapter begins with a study of existing methods where it is decided that using a color model that separates color into intensity and chromaticity components will constitute the basis of the shadow detection algorithm. An overview of the color spaces, HSV, IHSL, CIELAB, YCrCb, as well as a color model proposed in the literature, is given. The chapter ends with some tests and a discussion of the results.

### 3.1 Introduction

As can be seen from the examples in the previous chapter, all of the methods for background modeling will classify shadows as foreground, which is the expected behavior since shadows significantly differ from the background. However, in order for the segmented foreground to be useful for tracking objects, shadows need to be suppressed or at least detected. Shadows detected as foreground can cause several problems when extracting and labeling objects, two examples are object shape distortion and several objects merging together.

The purpose of this chapter is to evaluate some existing methods for shadow detection as well as choosing a method to use together with one of the background modeling algorithms in the previous chapter as a solution to the main objective, which is the topic of the next and final chapter.

### 3.2 Study of Existing Methods

After searching for material on methods for shadow detection, shadow removal, and shadow identification and reading many abstracts, the articles [4, 7, 9, 10, 11, 16] were chosen to constitute the basis of this study.

The most interesting article is perhaps Prati et al. [16] who presented a comprehensive survey of methods to detect moving shadows in 2003. They classified 21 existing methods into four groups and selected four methods from the various groups to implement and compare. The authors note that a model-based approach achieves the best results, but is often very complex and time consuming. For general-purpose shadow detection the

authors recommend their own proposed method based on exploiting the color information in the HSV color space.

Two of the more recent papers are Blauensteiner et al. 2006 [4] who use, what they call, an improved version of HSV to detect shadows and Martel-Brisson and Zaccarin 2005 [11] who use YUV color space in a similar fashion. Horprasert et al. 1999 [7] use their own color model, but the principal is the same, convert the RGB pixel values to a color model which separates intensity and chromaticity.

Khan and Reinhard 2004 [9] presented a survey of suitable color spaces for shadow edge detection while Kumar et al. 2002 [10] presented a survey of suitable color spaces for foreground and shadow detection for traffic monitoring systems. There is some overlap on the color spaces surveyed, but Khan and Reinhard conclude that a color opponent space such as CIELAB and  $L\alpha\beta$  performs significantly better than the other color spaces tested while Kumar et al. conclude that YCrCb gives fairly good results compared to the other color spaces tested. Unfortunately, YCrCb is not part of the survey by Khan and Reinhard while at the same time CIELAB and  $L\alpha\beta$  is not part of the survey by Kumar et al.

Due to the fact that so many methods are based on using a color representation that separates color into intensity and chromaticity components a method for shadow detection based on one of these color representations will be used in the final algorithm. No specific color space or color model will be chosen at this point since the articles do not agree on any color representation being the most suitable. Instead, the various color spaces, along with the color model used in [7], will be reviewed and discussed.

### 3.3 Color and Color Spaces

When discussing color spaces some commonly used terms often come up, this section presents those terms and their meaning before discussing the color spaces HSV, IHSL, CIELAB, YCrCb and the color model proposed in [7].

#### 3.3.1 Terminology

The definitions of the terms in the following section have been agreed upon and standardized by the *CIE - International Commission on Illumination*<sup>1</sup> which is an international organization devoted to the cooperation and exchange of information among its member countries on all matters relating to the science and art of lighting<sup>2</sup>. The international lighting vocabulary released by the CIE in 1987 is rather expensive; fortunately, Charles Poyntons FAQ about color [14] and Wyszecki and Stiles [19] gives some insight to the definitions relevant for our purposes.

**Color** – The perceptual result of light in the visible region of the spectrum, wavelengths in the range 400 – 700 nm, incident on the retina of the human eye.

Physical power, radiance, is expressed in a spectral power distribution SPD. The human retina has four types of cells that act as photoreceptors, three color sensitive (cone cells) and one intensity sensitive (the rod cell) which is only active at very low light levels. This is the motivation for using three numerical components to represent color.

**Brightness** – Attribute of visual sensation according to which an area appears to emit more or less light. Measured subjectively and has no units of measurement.

<sup>1</sup>The acronym is from the French title Commission Internationale de l'Eclairage.

<sup>2</sup>CIE homepage - <http://www.cie.co.at/cie/>.

**Intensity** – Radiant intensity, power per steradian (SI-unit:  $\frac{W}{sr}$ ).

**Luminance** – The luminous intensity per unit surface area, measured in candela per square meter ( $\frac{cd}{m^2}$ ). Luminous intensity (SI-unit:  $cd$ ) is the radiant intensity weighted by the spectral response of the human visual system.

The international recommendation for the high definition television standard, ITU Rec. 709, specifies the following weights to compute the luminance  $Y$  from the red, green, and blue color components

$$Y = 0.2126R + 0.7152G + 0.0722B . \quad (3.1)$$

**Lightness** – A measurement which takes into account the non-linear perceptual response of the human visual system.

A source having a luminance of only 18% of a reference luminance appears perceptually about half as bright.

**Hue** – Hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors, red, yellow, green, and blue, or a combination of two of them.

If the dominant wavelength of an SPD shifts, the hue of the associated color will shift.

**Saturation** – Saturation is the colorfulness of an area judged in proportion to its brightness.

The more an SPD is concentrated at a specific wavelength, the more saturated the associated color will appear. Saturation ranges from neutral gray through pastel to saturated color.

**Chroma** – Chroma is the attribute of a visual sensation which permits a judgment to be made of the degree a chromatic stimulus (possessing hue) differs from an achromatic stimulus (devoid of hue) of the same brightness.

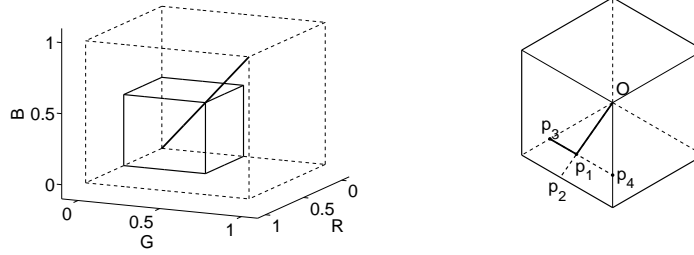
### 3.3.2 Hue, Saturation, Value – HSV

The HSV color space separates color into three components, hue, saturation, and value. It was introduced in [17] and developed for use in computer graphics to specify color in way more closely related to how we humans perceive color than when specified in terms of red, green, and blue.

The HSV color space transforms the RGB-cube into a cylinder in two steps. First, the three outermost surfaces of the cube defined by the diagonal line connecting the origin  $O = (0, 0, 0)$  and a point  $AC$  on the achromatic axis  $R = G = B$  is projected along the achromatic axis on a plane perpendicular to the axis, thus defining a hexagon, see figure 3.1.

Let  $P_1$  in figure 3.1 be the projection of a point  $P$ . The value  $V$  is defined as the maximum value of the RGB-values, the saturation  $S$  is defined as the ratio of the length of the vectors  $OP_1$  and  $OP_2$ , and the hue  $H$  is defined as the ratio of the length of the vectors  $P_3P_1$  and  $P_3P_4$ . Note that, if the point falls in any of the other sextants the hue is defined similarly but a little different. Since the saturation is normalized with respect to  $V$ , the hexcone is transformed into a cylinder.

Conversion to HSV from RGB is done by



**Figure 3.1.** RGB to HSV transformation process. Left, RGB subcube. Right, the three outermost surfaces of the subcube projected onto a plane perpendicular to the achromatic axis.

$$V = \max(R, G, B) , \quad (3.2)$$

$$S = \begin{cases} \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B)} & \text{if } \max(R, G, B) \neq 0 , \\ 0 & \text{otherwise,} \end{cases} \quad (3.3)$$

$$H = \begin{cases} \text{Undefined} & \text{if } S = 0 , \\ \frac{G-B}{\max(R, G, B) - \min(R, G, B)} & R = \max(R, G, B) , \\ 2 + \frac{B-R}{\max(R, G, B) - \min(R, G, B)} & G = \max(R, G, B) , \\ 4 + \frac{R-G}{\max(R, G, B) - \min(R, G, B)} & B = \max(R, G, B) . \end{cases} \quad (3.4)$$

Values will fall in the ranges  $V \in [0, 1]$ ,  $S \in [0, 1]$ , and  $H \in [0, 6]$ . The hue is multiplied by 60 to get a value in degrees. For the inverse transformation see [17].

It should be noted that the HSV color space is an attempt to specify color in a more natural way; there is no underlying physics or empirical data to motivate its form. It has been adopted by software vendors and is found in most graphics editing applications.

One effect of the normalization of the saturation is that color values that are not saturated, e.g. colors in dark regions, can have maximum saturation. Saturation in HSV has a different meaning than saturation as defined by the CIE.

For image and video processing applications there are some drawbacks using HSV. The hue for values close to the achromatic axis will be more or less unstable depending on the noise introduced by the capturing device. Also, both the saturation and hue components will be unreliable in dark regions due low signal to noise ratio. Another drawback is that the saturation depends on the value.

### 3.3.3 Improved Hue, Saturation, Lightness – IHSL

Noting the drawbacks of the HSV and other similar color spaces for image processing applications an alternative 3D-polar color representation was presented in [6]. As with HSV, the color is described using the three components, hue, saturation, and lightness. Lightness, or lightness function, is in this case a more general term for value. The saturation is independent of the lightness function used and the hue is defined as in HSV.

Conversion to IHSL from RGB using (3.1) as the lightness function is done by calculating



$$Y = 0.2126R + 0.7152G + 0.0722B , \quad (3.5)$$

$$S = \max(R, G, B) - \min(R, G, B) , \quad (3.6)$$

$$H = \begin{cases} \text{Undefined} & \text{if } R=G=B , \\ 2\pi - \arccos\left(\frac{R-G/2-B/2}{\sqrt{R^2+G^2+B^2-RG-RB-BG}}\right) & \text{if } B > G , \\ \arccos\left(\frac{R-G/2-B/2}{\sqrt{R^2+G^2+B^2-RG-RB-BG}}\right) & \text{otherwise.} \end{cases} \quad (3.7)$$

Values will fall in the ranges  $Y \in [0, 1]$ ,  $S \in [0, 1]$ , and  $H \in [0, 2\pi]$ . The inverse transformation for this case is given in [6].

Compared to HSV there are some advantages to using IHSL. One advantage is that a better suited lightness function can be used, the value  $V$  used in HSV conflicts badly with our perception of lightness. Another advantage is that the meaning of the term saturation is more comparable with CIE's definition. However, as with any color space defining a hue as above, it suffers from it being undefined for RGB-values on the achromatic axis.

### 3.3.4 Perceptually Uniform Color Space – CIELAB

The CIELAB color space is designed to be perceptually uniform, meaning that the euclidean distance between two points anywhere in the color space should be similar to the difference in color an observer would perceive.

Like any CIE color space its transformation has its roots in CIE XYZ, which is designed to accommodate all colors perceivable by the CIE standard observer and is based on empirical data obtained in the 1920's and 1930's. An important property of the CIE XYZ is that it is a device independent color space, meaning that on a calibrated display device the colors shown will be the expected. For more information on CIE color spaces and the science of colorimetry see [19].

CIELAB separates color into lightness  $L^*$ , as defined in section 3.3.1, and two chromaticity components  $a^*$  and  $b^*$ . Converting an image from RGB to CIELAB is done via the CIE XYZ color space and information regarding the white point and primaries used when capturing the image must be known.

RGB-values conforming to the ITU Rec. 709 standard are converted to XYZ-values by the linear transformation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} . \quad (3.8)$$

The  $L^*a^*b^*$ -values are then computed using the equations

$$L^* = \begin{cases} 116 \left( \frac{Y}{Y_n} \right)^{\frac{1}{3}} - 16 & \text{if } \frac{Y}{Y_n} \geq 0.008856 , \\ 903.3 \frac{Y}{Y_n} & \text{otherwise,} \end{cases} \quad (3.9)$$

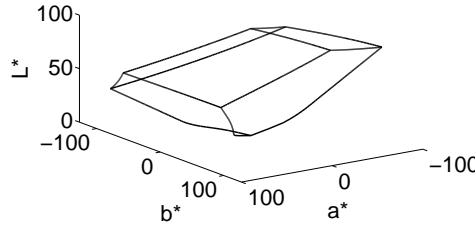
$$a^* = 500 \left[ f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] , \quad (3.10)$$

$$b^* = 200 \left[ f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right] , \quad (3.11)$$

where

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{if } t \geq 0.008856, \\ 7.787t + \frac{16}{116} & \text{otherwise.} \end{cases} \quad (3.12)$$

$X_n, Y_n, Z_n$  are obtained by setting  $R_{709} = G_{709} = B_{709} = 1$  in 3.8.  $L^*$  will be in the range  $[0, 100]$  while  $a^*$  and  $b^*$  will be in the range  $[-127, 128]$ . The shape of CIELAB depends on the calibration parameters used, for instance, using ITU Rec. 709 parameters the space will have the shape as shown in figure 3.2.



**Figure 3.2.** Shape of the ITU Rec. 709 RGB-cube in CIELAB.

### 3.3.5 Color Space Used for Image and Video Encoding – YCbCr

As with CIELAB the YCbCr color space separates color into a luminance component and two chromaticity components. Detailed information regarding this color space is given in [15]. The main reason for the existence of the YCbCr color space is a historical one, when color TVs were introduced there was a need to be backwards compatible with black and white TVs, and the separation of luminance and chrominance made this possible. The human visual system is far less sensitive to errors in chromaticity than luminance; allowing for less bandwidth to be used to transmit the chromaticity information. It is still used today because of this, along with the fact that conversion to and from RGB is very simple.

Conversion to YCbCr is done via YPbPr which denotes YCbCr prior to offset and scaling for digital storage. The following transformation is used to convert gamma corrected<sup>3</sup> ITU Rec. 709 RGB-values  $R', G', B'$  to Y'PbPr-values

$$Y' = 0.2126R' + 0.7152G' + 0.0722B' , \quad (3.13)$$

$$Pb = 0.5389(B' - Y') , \quad (3.14)$$

$$Pr = 0.6350(R' - Y') , \quad (3.15)$$

where  $Y' \in [0, 1]$ ,  $Pb \in [-0.5, 0.5]$ , and  $Pr \in [-0.5, 0.5]$ , and the gamma correction is done using

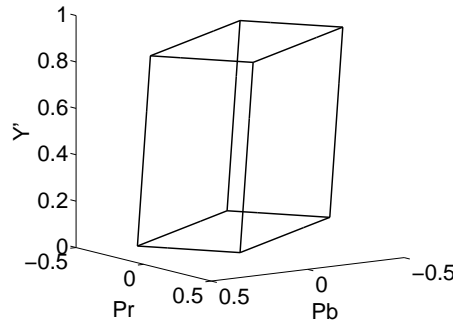
<sup>3</sup>Gamma correction refers to compensating for the nonlinearity of CRTs in order to achieve correct reproduction of relative luminance, see [15] for details.

$$R' = 1.099R^{0.45} - 0.099 , \quad (3.16)$$

$$G' = 1.099G^{0.45} - 0.099 , \quad (3.17)$$

$$B' = 1.099B^{0.45} - 0.099 . \quad (3.18)$$

As with CIELAB the transformed RGB-cube will be an irregular shape, see figure 3.3.



**Figure 3.3.** Shape of the ITU Rec. 709 RGB-cube in  $Y'PbPr$ .

As offset and scaling are of no importance for the purpose of shadow detection,  $Y'PbPr$  will be used for the remainder of this thesis.

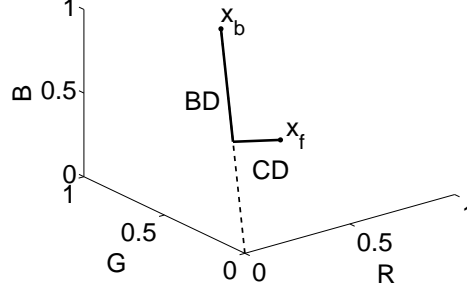
### 3.3.6 Color Model of Horprasert et al.

In contrast to the color spaces described above, the color model proposed by Horprasert et al. in [7] does not specify any conversion formulas. Since the purpose of the color model is to detect shadow points by comparing an image to a background image, the model is presented as way to calculate, what the authors call, brightness and chromaticity distortion. Each pixel value in the RGB color space is assumed to lie on a *chromaticity line*, which connects the pixel value and the origin. Roughly speaking, the brightness distortion is the deviation of a pixel value in the background image along this line, and the chromaticity distortion is pixel values deviation perpendicular to the line, see figure 3.4.

The model is based on the empirical observation that shadowed pixel values tend to follow straight lines connecting the values and the origin in the RGB color space.

## 3.4 Comparison

Considering the case of detecting shadow points by comparing an image to a known background image, methods for the different color representations are given in this section, followed by two tests evaluating their performance.



**Figure 3.4.** Brightness and chromaticity distortion (BD and CD) in the RGB color space,  $x_b$  and  $x_f$  denotes background and foreground values respectively.

### 3.4.1 Methods for Evaluation

The color spaces HSV and IHSL are compared using the same hypothesis, which is that a shadowed pixel value's value and saturation will decrease while the hue remains relatively constant. Let  $x_f, x_b$  denote a foreground and background pixel value respectively, they can be RGB-vectors, HSV-vectors, or something else depending on the context. Also, let  $sp$  denote a shadow mask of the same size as the images. The hypothesis is tested by evaluating

$$sp = \begin{cases} 1 & \text{if } \alpha \leq \frac{x_{f,L}}{x_{b,L}} \leq \beta \\ & \wedge x_{f,S} - x_{b,S} \leq \tau_S \\ & \wedge |x_{f,H} - x_{b,H}| \leq \tau_H, \\ 0 & \text{otherwise,} \end{cases} \quad (3.19)$$

where  $L$  denotes the lightness function and is  $V$  for HSV and  $Y$  for IHSL. This is the same test as in [16] but not in [4], however, their hypothesis is the same.

CIELAB and Y'PbPr can be compared in a similar manner under the hypothesis that the lightness or luminance component should decrease and the chromaticity coordinates should remain relatively constant. Using the same notation as above,  $sp$  can be expressed as

$$sp = \begin{cases} 1 & \text{if } \alpha \leq \frac{x_{f,L}}{x_{b,L}} \leq \beta \\ & \wedge (x_{f,c_1} - x_{b,c_1})^2 + (x_{f,c_2} - x_{b,c_2})^2 \leq \tau_C, \\ 0 & \text{otherwise,} \end{cases} \quad (3.20)$$

where  $L$  denotes the lightness function and is  $L^*$  for CIELAB and  $Y'$  for Y'PbPr, and  $c_1, c_2$  denotes the chromaticity coordinates and is  $a^*, b^*$  for CIELAB and  $Pb, Pr$  for Y'PbPr. While not explicitly stated in [9] or [10], equation (3.20) is meant to reflect and test the hypothesis formulated therein.

Finally, using the color model from Horprasert et al.  $sp$  can be expressed as

$$sp = \begin{cases} 1 & \text{if } \alpha \leq \frac{x_f \cdot \hat{x}_b}{\|\hat{x}_b\|} \leq \beta \\ & \wedge \|x_f - (x_f \cdot \hat{x}_b)\hat{x}_b\| \leq \tau_C, \\ 0 & \text{otherwise,} \end{cases} \quad (3.21)$$

where  $\hat{x}$  denotes that the vector is normalized.

### 3.4.2 Test 1

To get an idea of how well these methods work, a test on real data has been carried out. The same image as in the examples in chapter 2 was used, and the background image was taken to be the most probable background image when using mixture models for background modeling, see figure 3.5. Even though it has uncertainty to it, it will be considered the correct background image here. The parameters used were tweaked to give best visual results and are given in table 3.1. A test like this is of course very subjective and case dependant, and only interesting to get an idea of how the methods perform.

**Table 3.1.** Parameter settings, chosen to give best visual results.

HSV	$\alpha = 0.3$	$\beta = 0.9$	$\tau_S = 0.3$	$\tau_H = 0.5$
IHSL	$\alpha = 0.3$	$\beta = 0.9$	$\tau_S = 0.3$	$\tau_H = \pi$
CIELAB	$\alpha = 0.3$	$\beta = 0.9$	$\tau_C = 15$	
Y'PbPr	$\alpha = 0.3$	$\beta = 0.9$	$\tau_C = 0.07$	
Horprasert	$\alpha = 0.3$	$\beta = 0.9$	$\tau_C = 0.1$	

Figure 3.6 shows that using HSV, IHLS, or Horprasert's color model gives very similar results, and using CIELAB or Y'PbPr also gives a very similar result. Closer inspection of the data reveals that the hue component in both HSV and IHSL contribute very little to the discrimination of shadow points. In fact, since the asphalt is gray the RGB pixel values lies close to the achromatic axis, which together with noise introduced by the imaging equipment causes the hue to be unstable and unreliable. The chromaticity components of CIELAB and Y'PbPr also contribute very little to the discrimination of shadow points. As a result the brightness component is really the only information used to detect shadow points, which is why the dark car in the center of the image is completely detected as shadow when using these color spaces.

### 3.4.3 Test 2

As an attempt to evaluate the methods more objectively the following test has been carried out. The left image in figure 3.7 was taken indoors. Using the same camera settings and lighting conditions an image of the background was taken. Ground-truth data was generated by manually segmenting the objects. The segmented foreground was calculated by evaluating  $\|x_f - x_b\| > 0.2$ . All foreground points that do not correspond to object points are considered shadow points.

Using the ground-truth data together with the segmented foreground, false positives  $FP$  and false negatives  $FN$  were calculated, where false positives refers to background/foreground points classified as shadow points and false negatives refers to shadow points classified as background/foreground.  $FP$  and  $FN$  is calculated in percent using the true number of shadow and object points.

Judging from the data in table 3.2 using Horprasert's color model gives the best results in terms of the least false positives while IHSL gives the best results in terms of least false negatives, however, the large amount of false positives is significant. The best performing methods are HSV and Horprasert while Y'PbPr can be a good alternative if the application requires a low amount of false negatives. A problem with a test like this is that objects' reflected light will affect the background. The false negatives in HSV, CIELAB, Y'PbPr, and Horprasert in the green block's cast shadow could very well be

**Table 3.2.** False Positives/Negatives.

	False Positives (%)	False Negatives (%)
HSV	2.66	16.7
IHLS	41.48	3.55
CIELAB	14.30	20.19
Y'PbPr	15.70	6.01
Horprasert	1.11	14.21

caused by the reflected light from the yellow block, see figure 3.8. This problem can be avoided by using scenes with only one object present at a time. It is an interesting observation though, since in real scenes several objects will be present and their reflected light will interact with the background.

### 3.4.4 Test 3

The purpose of the third and final test is to see how well the methods perform by optimizing the parameters to minimize false positives and false negatives using the images in test 2. Some time was spent on each method choosing parameter values by looking at the underlying data, i.e.  $\frac{x_{f,L}}{x_{b,L}}$  and  $x_{f,S} - x_{b,S}$  in (3.19)-(3.21) etc., trying to find the best possible parameter values. Using the parameters in table 3.3 the results in table 3.4 were obtained.

**Table 3.3.** Parameter settings, optimized to minimize false positives and false negatives.

HSV	$\alpha = 0.2$	$\beta = 0.95$	$\tau_S = 0.3$	$\tau_H = 0.1$
IHSL	$\alpha = 0.3$	$\beta = 0.9$	$\tau_S = 0.1$	$\tau_H = 0.3\pi$
CIELAB	$\alpha = 0.3$	$\beta = 0.9$	$\tau_C = 12$	
Y'PbPr	$\alpha = 0.2$	$\beta = 0.95$	$\tau_C = 0.07$	
Horprasert	$\alpha = 0.3$	$\beta = 0.9$	$\tau_C = 0.11$	

**Table 3.4.** False Positives/Negatives using optimized parameters.

	False Positives (%)	False Negatives (%)
HSV	2.19	14.62
IHLS	21.84	4.29
CIELAB	10.41	20.64
Y'PbPr	15.86	4.12
Horprasert	2.83	10.64

The performance of HSV, Y'PbPr, and Horprasert is very similar to the results in test 2. For HSV and IHSL the hue component did help to detect some shadow points in this case. The large amount of false positives in IHSL is much less but still very high. Tweaking  $\alpha$  and  $\beta$  in Y'PbPr gives a small improvement. As for Horprasert, it was

difficult to get better results than in test 2, increasing  $\tau_C$  gave fewer false negatives but also more false positives.

Using the new parameter values on the image pair in test 2 gives the result in figure 3.12. From the figure it is clear that the choice of parameter values is very important in HSV and IHSL. Both CIELAB and Y'PbPr perform similar to test 1 which is not surprising since they both mainly use their brightness component to discriminate shadow points. The performance in Horprasert is not very surprising either since the parameters were barely changed.

## 3.5 Discussion

It is difficult to carry out any truly objective and meaningful tests to evaluate the performance of the methods presented in this chapter. Manually segmenting the object and shadow points introduces uncertainty. Trying to simulate an object and its cast shadow using 3D graphics will be of minor interest as a known color model which tries to simulate a real shadow is used. The best, or most useful, test is probably to apply the methods on several video sequences and simply view the result and subjectively decide which methods perform sufficiently well for the application at hand.

In the tests presented here Horprasert proved to be best method for the application of traffic monitoring. While shadow detection in HSV can give very similar results, the simple parameter choice for Horprasert's method is an advantage. Also, Horprasert's color model does not suffer from an unstable hue for RGB-values close to the achromatic axis, which is a practical advantage.

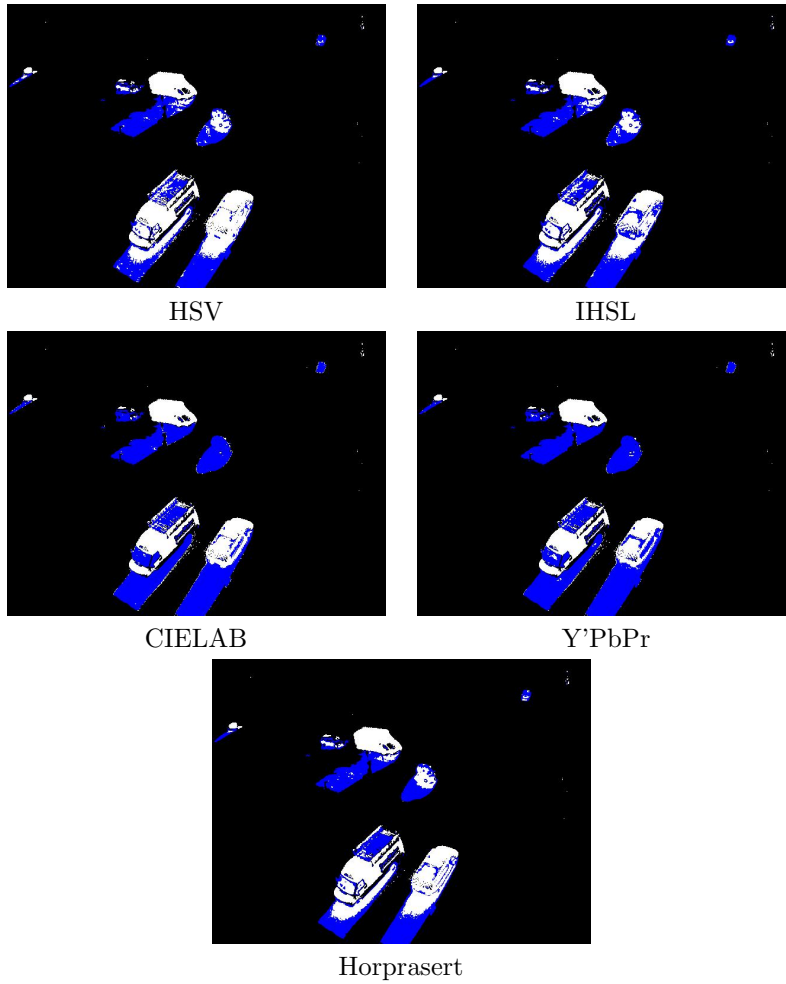
Any method that exploits color theory to detect shadow points is limited in performance since light interaction is a very complex process and difficult to model. Assuming that pixel values follow chromaticity lines is a generalization that works well in determining darkened pixel values due to the occlusion of light sources.

As a side note it should be mentioned that an informal test with the purpose of getting an idea of how accurate this assumption is has been carried out. Using various color cards, printed with a high quality printer, objects were placed so that their cast shadow fell onto the cards. Images were taken with a digital camera, and 3D scatter plots of areas containing the cast shadow points show that for achromatic colors the assumption seems to be valid while for chromatic colors, such as saturated red, yellow, green etc., the shadow points seem to follow curved instead of straight lines.

To conclude, Horprasert's color model will be used for shadow detection in the final algorithm.

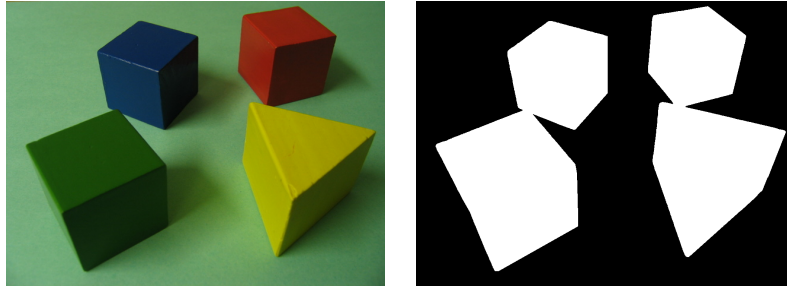


**Figure 3.5.** Image and the most probable background image.

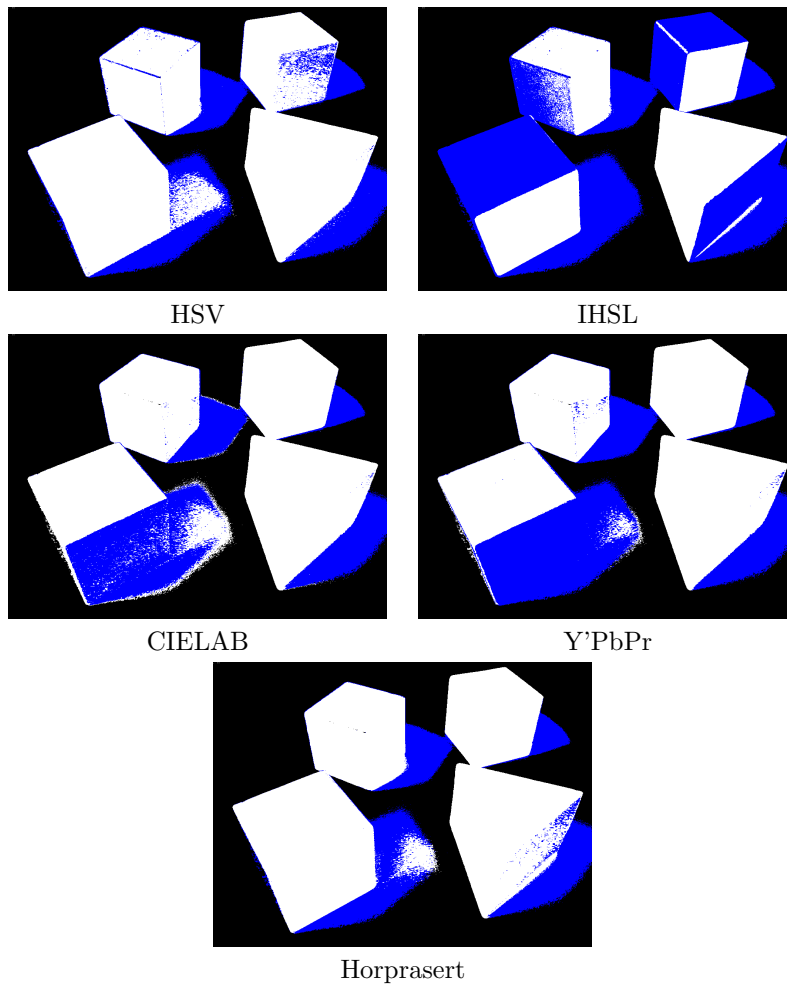


**Figure 3.6.** Shadow detection exploiting color information using the parameter settings in table 3.1.

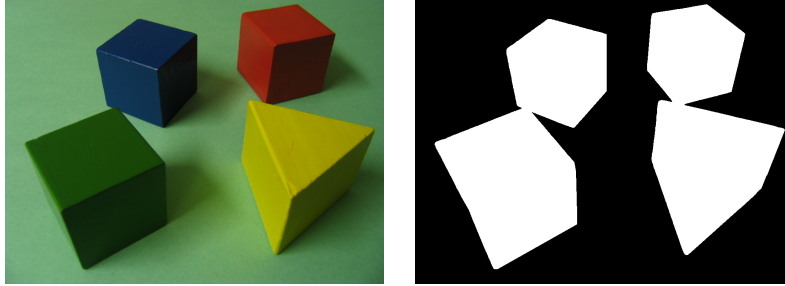




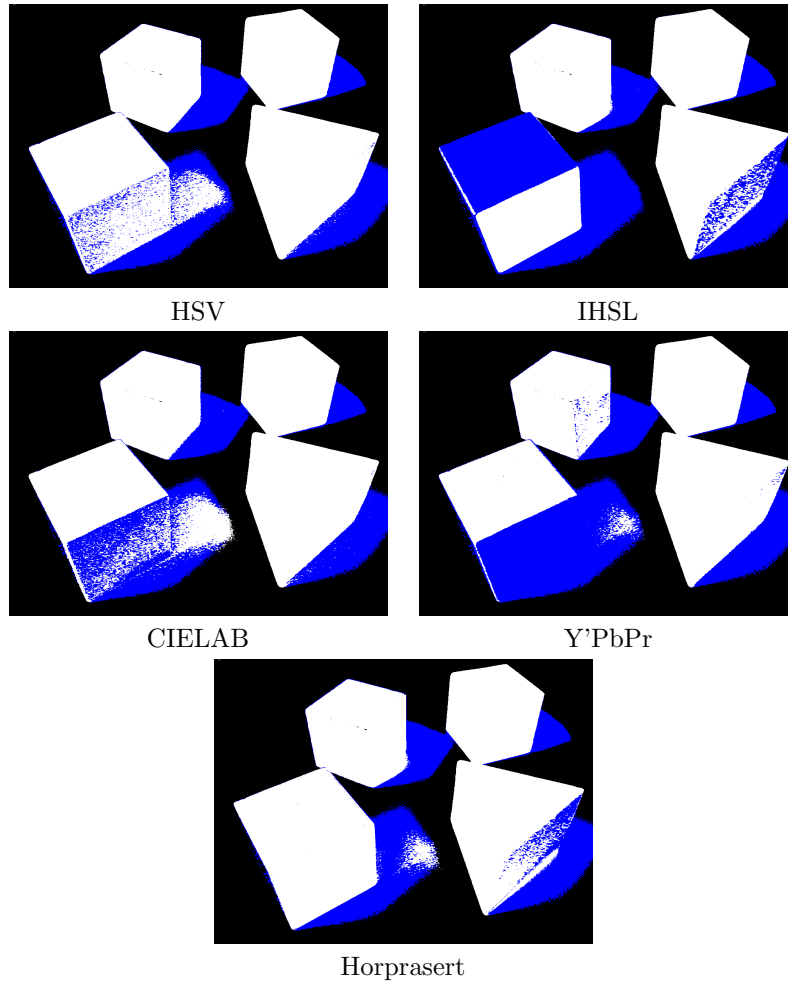
**Figure 3.7.** Original image and manually segmented ground-truth data.



**Figure 3.8.** Shadow detection exploiting color information using the same parameters as in test 1.



**Figure 3.9.** Original image and manually segmented ground-truth data.



**Figure 3.10.** Shadow detection exploiting color information using the parameters in table 3.3.



Figure 3.11. Image and corresponding background image.

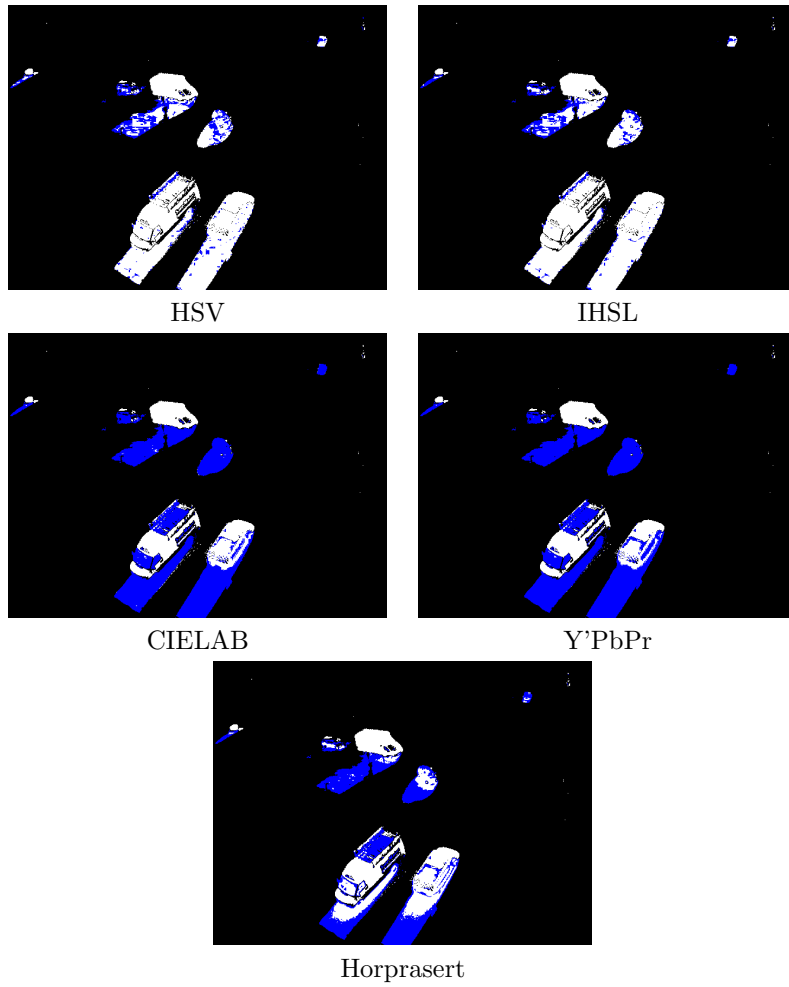


Figure 3.12. Using the parameter values optimized for test 3 on test 1.



## Chapter 4

# Proposed Method

This chapter proposes a solution to the main objective of the thesis. The main focus is the final algorithm and its implementation. Pseudo code and an example of typical results with an accompanying discussion are given. The chapter ends with some conclusions and a summary of the ideas for future work that have been mentioned in the thesis.

### 4.1 Introduction

The main objective of this thesis is to propose a method for background modeling that incorporates the detection and removal of shadows. The purpose of this chapter is to propose a solution to the main objective as well as to give pseudo code of a possible implementation and discuss the performance and limitations of such an implementation.

### 4.2 Proposed Method

As a solution to the main objective of this thesis, I propose using Stauffer and Grimson's algorithm, slightly modified as described in chapter 2, for background modeling together with a shadow detection algorithm based on Horprasert's color model.

The main reasons for choosing Stauffer and Grimson's algorithm is that the Gaussian mixture model representation of the scene statistics has proven to be very flexible and reasonably efficient when implemented. Channel encoding works quite well too, but the method presented in chapter 2 is not as efficient as Stauffer and Grimson's algorithm on larger images, especially in terms of memory usage.

As concluded in the previous chapter Horprasert's color model will be used to detect shadow points, that is shadowed pixels are assumed to follow straight lines connecting the pixel value and the origin in the RGB color space. The main reasons for choosing this color model is that it provided good results on all test scenes in the comparison without having to tweak the parameters very much.

### 4.3 A Complete Implementation

To implement the method in its entirety only the background segmentation algorithm 3 needs to be changed, the algorithm for updating the background model, algorithm 2, can be used as is. However, using a couple of the practically useful tweaks mentioned in chapter 2, not renormalizing the mixing densities to allow modes to die out and setting a lower limit on the variance to avoid overfitting, the algorithm can be written as shown in algorithm 6. The segmentation algorithm can be written as in algorithm 7. Note that  $x_{rgb,t}$  and  $\mu_{rgb,k}$  denotes the RGB-components of  $x_t$  and  $\mu_k$ . Shadow detection can be used together with other properties in the same input vector as long as it's clear which components correspond to red, green, and blue. The notation used in both algorithms can be found in table 4.1.

**Table 4.1.** Notation for algorithm 6 and 7.

$x_t$	pixel value at time t
$f_t$	frame at time
$w$	mixing density
$\mu$	mean
$\sigma^2$	variance
$K$	number of mixture components
$D$	dimension of mixture components
$\alpha$	learning rate
$\lambda$	threshold
$\sigma_{init}^2$	initial covariance vector
$\sigma_{min}^2$	minimum covariance component allowed
$T$	threshold
$k$	integer in the range $[1, \dots, K]$
$m$	integer in the range $[1, \dots, K]$
$match$	boolean
$B$	background model
$\hat{B}$	binary background image
$\beta_1, \beta_2$	darkness thresholds
$\tau_c$	chrominance threshold
$sp$	shadow mask
$\circ$	element-wise multiplication

---

**Algorithm 6** Background modeling using a mixture of D-dimensional Gaussians

---

```

1: # Update Model
2: for all  $x_t$  in  $f_t$  do
3:   for all  $k \in [1, \dots, K]$  do
4:      $d_k^2 = \sum_{d=1}^D \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$ 
5:     if  $d_k < \lambda$  then
6:       if  $match = 0$  then
7:          $m = k$ 
8:       else if  $\frac{w_k}{\sqrt{\|\sigma_k^2\|}} > \frac{w_m}{\sqrt{\|\sigma_m^2\|}}$  then
9:          $m = k$ 
10:      end if
11:       $match = 1$ 
12:    end if
13:  end for
14:
15:  if  $match = 0$  then
16:     $m = K$ 
17:     $w_m = \alpha$ 
18:     $\mu_m = x_t$ 
19:     $\sigma_m^2 = \sigma_{init}^2$ 
20:  else
21:     $w_m = (1 - \alpha)w_m + \alpha$ 
22:     $\rho_m = \frac{\alpha}{w_m}$ 
23:     $\mu_m = (1 - \rho_m)\mu_m + \rho_m \cdot x_t$ 
24:     $\sigma_m^2 = (1 - \rho_m)\sigma_m^2 + \rho_m(x_t - \mu_m) \circ (x_t - \mu_m)$ 
25:
26:    Make sure no components of  $\sigma_m^2$  are less than  $\sigma_{min}^2$ .
27:  end if
28:
29:  for all  $k \in [1, \dots, K] \neq m$  do
30:     $w_k = (1 - \alpha)w_k$ 
31:  end for
32:
33:  if  $match \neq 0$  then
34:    Sort  $w, \mu, \sigma$  with respect to  $(\frac{w_1}{\sqrt{\|\sigma_1^2\|}}, \dots, \frac{w_K}{\sqrt{\|\sigma_K^2\|}})$ .
35:  end if
36:
37:   $B = argmin_b \left( \sum_{k=1}^b w_k > T \right)$ 
38: end for

```

---

---

**Algorithm 7** Background segmentation for algorithm 6 incorporating shadow detection.

---

```

1: # Shadow detection.
2: for all  $x_t$  in  $f_t$  do
3:    $\hat{B} = 0$ 
4:   for all  $k \in [1, \dots, B]$  do
5:      $d_k^2 = \sum_{d=1}^D \frac{(x_{t,d} - \mu_{k,d})^2}{\sigma_{k,d}^2}$ 
6:     if  $d_k < \lambda$  then
7:        $\hat{B} = 1$ 
8:     end if
9:   end for
10:
11:  if  $\hat{B} = 1$  then
12:    for all  $k \in [1, \dots, B]$  do
13:       $Dv = \frac{x_{rgb,t}^T \hat{\mu}_{rgb,k}}{\|\mu_{rgb,k}\|}$ 
14:       $Dc = \|x_{rgb,t} - (x_{rgb,t}^T \hat{\mu}_{rgb,k}) \hat{\mu}_{rgb,k}\|$ 
15:      if  $\beta_1 \leq Dv \leq \beta_2$  and  $Dc \leq \tau_c$  then
16:         $sp = 1$ 
17:        break
18:      else
19:         $sp = 0$ 
20:      end if
21:    end for
22:  end if
23: end for

```

---



## 4.4 Results and Discussion

As mentioned in the previous chapter it is difficult to conduct any meaningful tests to objectively evaluate the performance of the shadow detection algorithm, the same is true for the final algorithm as well. Even though objective tests are difficult to carry out at the time of writing, it is clear that the complete algorithm is useful to the tracking process in many cases, as long as the objects' color properties differ enough from the background's properties a large portion of the cast shadow is often detected correctly. To get an idea of how the complete algorithm performs two tests have been carried out. The purpose of the first test is to see how well the method performs on a long video sequence and note the methods general performance. The purpose of the second test is to identify problematic situations and give a qualitative measure of the methods performance.

### 4.4.1 A Three Hour Sequence

In the following test a three hour video sequence recorded midday in November at an intersection in Sweden was used as the input to the complete algorithm. Figure 4.2 shows the input and the output at 20 minute intervals. The current frame is shown to the left and the segmented foreground is shown to the right, white points correspond to object points, blue points (or gray points if printed in black and white) correspond to shadow points, and black points correspond to background points.

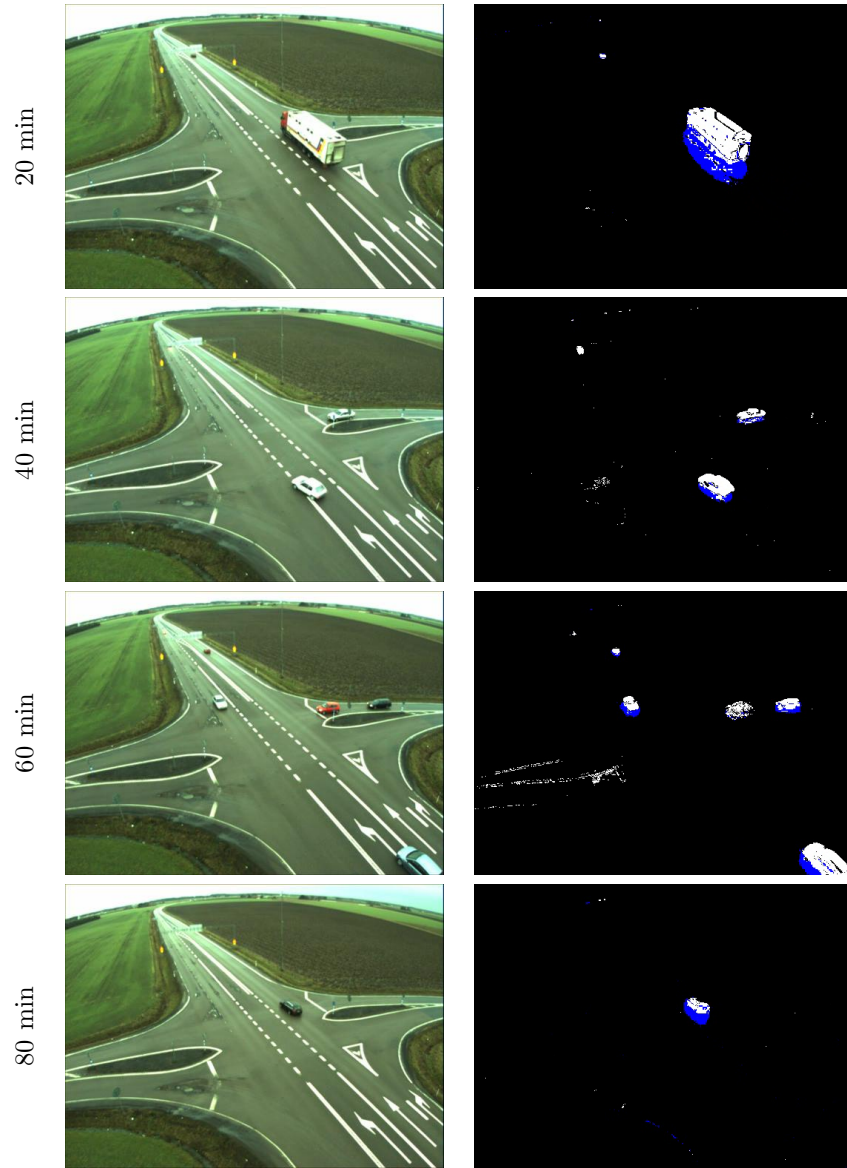
The learning rate  $\alpha$  was set so that a Gaussian's mixing density  $w_k$  will increase to  $T$  in 10 minutes if it repeatedly matches. To save processing time the background model was only updated two times per second,  $\alpha$  was chosen with this in mind. In these terms  $\alpha$  can be set using

$$\alpha = 1 - (1 - T)^{\frac{1}{n-1}}, \quad (4.1)$$

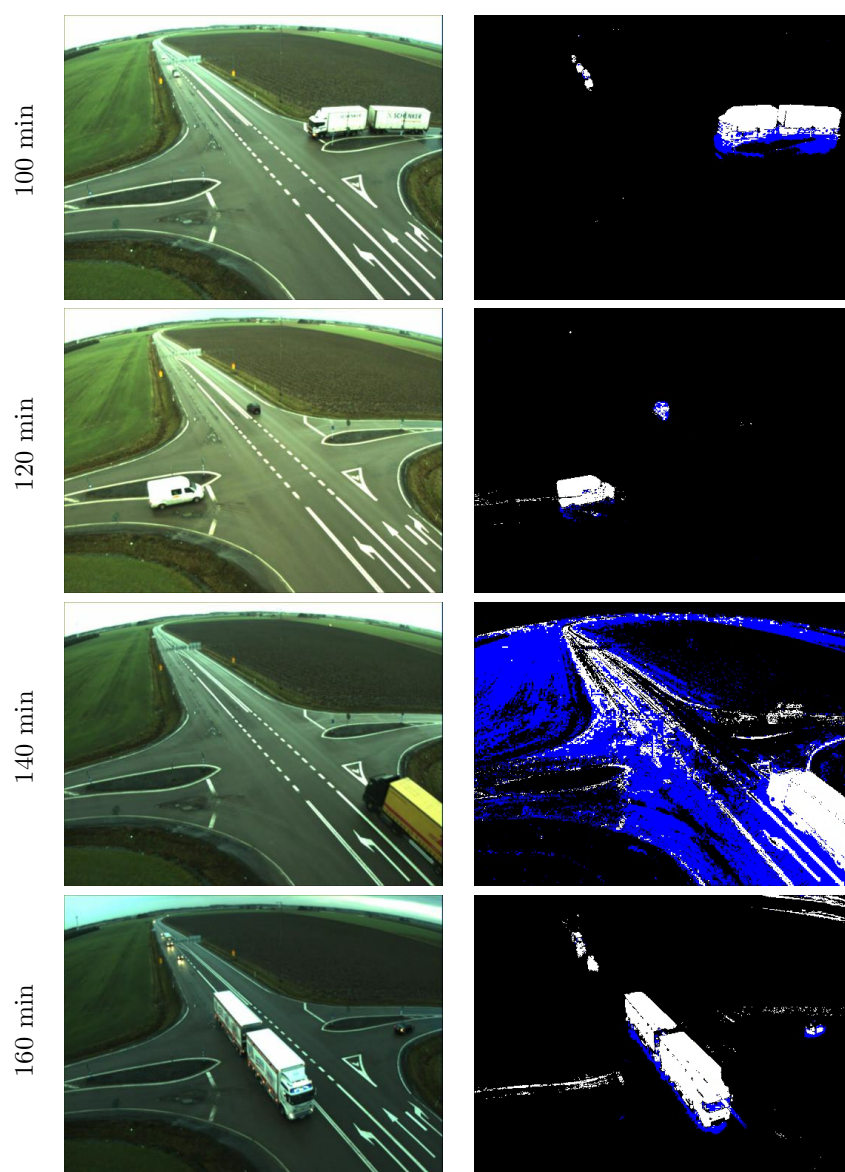
where  $n$  denotes the desired number of updates. So, in this case, 10 minutes of video at 20 frames per second and updating only two times per second corresponds to  $n = 1200$ , which yields  $\alpha \approx 0.0013$ . A complete list of the parameter values used is given in table 4.2.

**Table 4.2.** Parameter values used to generate figure 4.2.

Background Modeling		Shadow Detection	
$K$	= 3	$\beta_1$	= 0.3
$\alpha$	= 0.0013	$\beta_2$	= 0.95
$\lambda$	= 3.0	$\tau_c$	= 0.04
$T$	= 0.8		
$\sigma_{init}$	= 0.12		
$\sigma_{min}$	= 0.04		

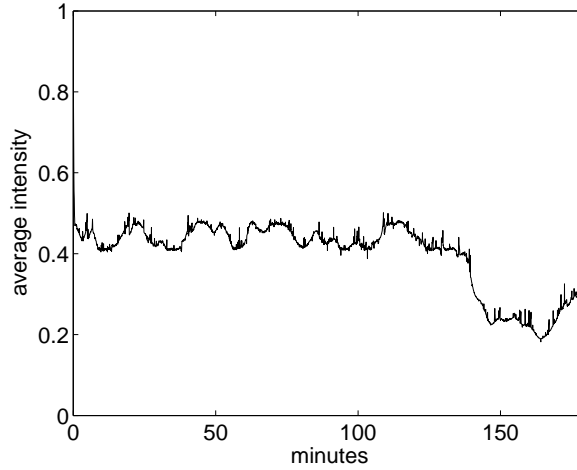


**Figure 4.1.** Results obtained when running the complete algorithm on a three hour video sequence recorded on a November afternoon in Sweden. The parameters used are given in table 4.2.



**Figure 4.2.** Results obtained when running the complete algorithm on a three hour video sequence recorded on a November afternoon in Sweden. The parameters used are given in table 4.2.

From figure 4.1 and 4.2 we can clearly see that the shadow detection is helpful. Remember that all blue points would be considered object points without the shadow detection, which would lead to severely distorted object shapes. The result after 140 minutes may seem erroneous and unwanted, but the learning rate was set to account for 10 minutes of data and the large amount of shadow points is probably due to the brightness level having significantly changed in less than 10 minutes. To verify this, figure 4.3 shows a plot of the mean value of every 60:th frame of the video sequence. The average intensity does infact drop about 50-60% in a short amount of time approximately 140 minutes into the sequence. Even though this type of behaviour may seem unwanted, it could be used as a tool to detect situations when the overall brightness rapidly decreases such as clouds covering the sun. Knowing when such situations occur allows one to dynamically change the learning rate so that an accurate background model can quickly be restored.



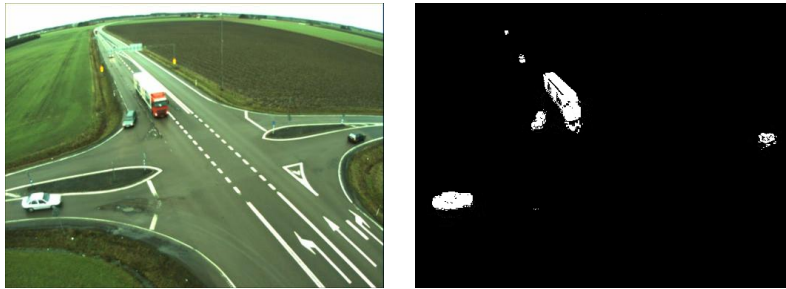
**Figure 4.3.** Average intensity per frame in video sequence in figure 4.2.

Setting the learning rate  $\alpha$  can be a bit tricky. The background model is designed to handle several modes per pixel, and the Gaussians with a combined mixing density greater than  $T$  are considered background modes. This means that stationary objects can be deemed background after relatively short periods of time even when the learning rate is set to a very low value. For example, consider a point in the image that is uni-modal and has a high mixing density, say  $w_k = 0.95$ . When a new object enters that point, the point will only be considered foreground until the background mode's mixing density decreases to less than  $T$ . In the example given here, using  $w_k = 0.95$ ,  $T = 0.8$ , and  $\alpha = 0.0013$ , a car that stops and waits to cross the intersection will be absorbed into the background after 133 updates if it remains stationary. Updating the model two times per seconds, 133 updates translates to approximately 2 minutes. This effect can be seen in the results after 60 minutes, the red car to the right is beginning to be absorbed into the background.

#### 4.4.2 A More Detailed Look

A test that would be interesting and fairly objective is to plug the complete algorithm into a complete tracking system and let the system run through a few selected video sequences for which ground-truth data have been manually determined. Then compare the number of objects that can be successfully tracked with and without the shadow detection. Unfortunately, the tracking system for which this work is intended is not yet in a state which allows for such a test to be carried out.

Using 10 minutes from the three hour video sequence (minutes 30-40) the number of vehicles that passed the scene was manually counted by viewing the video. Then it was determined if it would be possible to extract and label the vehicles correctly by viewing binary background/foreground image, subjectively estimating the future performance and accuracy of the tracking system. For example, in figure 4.4 the tracking system is assumed to be able to extract and label four vehicles correctly. Even though the tracking system would probably be able to extract the two cars in the upper left corner they are considered to be in a non-valid region of the image and are ignored. The reason for this is that the vehicles can be extremely small and often merge together in this region. Also, it's simply impossible to track vehicles as they drive into the horizon. The yellow markers slightly behind the truck are used to draw a line that separates the valid and non-valid regions.



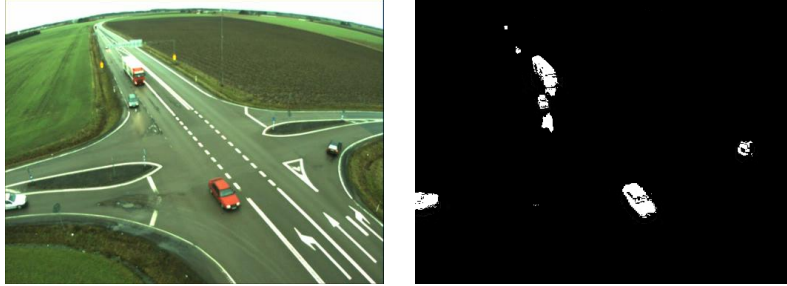
**Figure 4.4.** The tracking system is assumed to be able to extract and label four vehicles correctly, the two cars in the upper left corner are considered to be in a non-valid region of the image.

A total of 122 cars and trucks passed during the 10 minute clip. In the clip three problematic situations occurred. The most frequent problematic situation that occurred 8 times was that reflections from the vehicles head-lights caused a binary blob to appear slightly in front of the vehicles, that would be detected as a vehicle. Figure 4.5 shows an example of this situation.

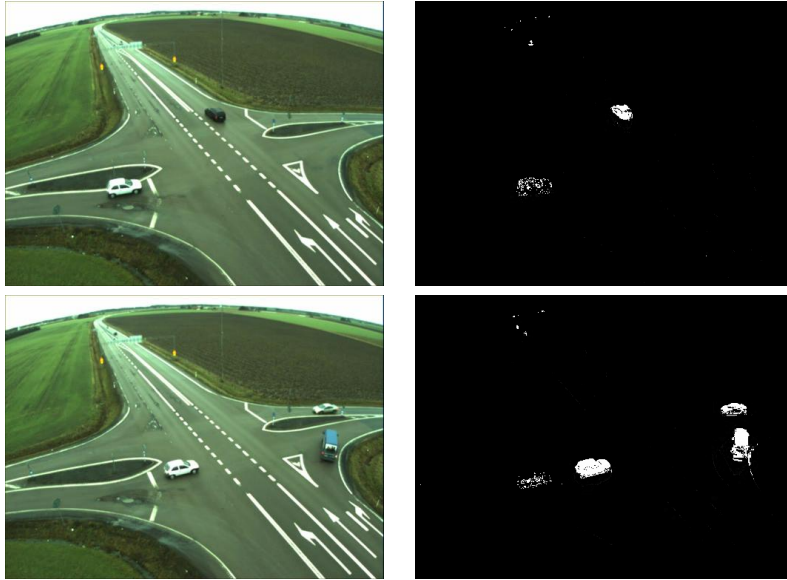
Horprasert et al. use their color model to detect highlighted points as well as shadow points which could provide the necessary solution. Note that this is only a problem when the road is wet. In this case the problem only occurs in a small region of the image, meaning that binary blobs only temporarily appear and may not be a big problem.

Another problematic situation that occurred once is that a vehicle stood still too long and was absorbed into the background model. As an unfortunate consequence the background model's previous mode's mixing density had decreased enough for the area to be considered foreground after the car left.

An adaptive background modeling technique is designed to dynamically adapt to changes in the scene, with time stationary objects should be deemed background. The



**Figure 4.5.** Vehicles head-lights cause binary blobs to appear slightly in front of the vehicles.



**Figure 4.6.** Top row: the white car is almost completely absorbed into the background. Bottom row: the car leaves trails when it drives off.

method as it is presented here uses a history of values to estimate a background model. When processing recorded data the binary background/foreground image can be segmented using a model from a later time, which may eliminate the trails.

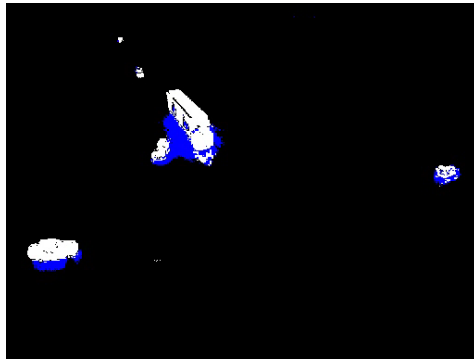
The third and final problematic situation that occurred during this 10 minute clip which is not a due to the proposed method, but rather a problematic situation of a more general nature, is that strong winds cause the camera to shake. Since this method in no way considers neighboring pixels' mixture models, the background is often deemed foreground near sharp edges, see figure 4.7.

This problem is probably best handled by stabilizing the video in a prior step.



**Figure 4.7.** Strong winds cause the camera to shake.

To end this test on a more positive note, figure 4.8 shows a car and a truck that is successfully separated thanks to the shadow detection.



**Figure 4.8.** A car and a truck is separated thanks to the shadow detection and can be extracted and labeled correctly.

To conclude, in the 10 minute clip at least 99% of the vehicles can be extracted and labled correctly, the biggest problem in this clip is the false positives caused by head-light reflections.

#### 4.4.3 Computational Performance

The software implementation of the method was mainly written in C/C++ utilizing Matlab's mex-interface to allow for easy data analysis and visualization. The video sequence used in the test presented here has a size of 512x384 pixels and a frame-rate of 20 fps. The background model was updated every tenth frame while the background/foreground segmentation was performed for every frame. The three hour video sequence took approximately 12 hours to process (5 fps) on an AMD Atholon 64 3800+ with one gigabyte of memory running Linux.

## 4.5 Conclusions and Ideas for Future Work

The proposed method provides an excellent basis for the extraction and labeling of objects. Using mixture models provides a flexible and powerful method for background modeling and the shadow detection algorithm is very helpful. Any method that exploits color theory to detect shadow points is limited in performance since light interaction is a very complex process and difficult to model. Assuming that pixel values follow chromaticity lines is a generalization that works well in determining darkened pixel values due to the occlusion of light sources. If better accuracy is needed a different approach is necessary. A shadow detection algorithm such as this can still be very useful, for example, it can be used as a first step in determining the presence and severity of shadows.

In the tracking system for which this work is intended, one of the necessary steps involves estimating objects three dimensional sizes. This is done by fitting three dimensional boxes around objects. Work has begun to simulate shadows under the assumption that segmented objects contain the objects and their cast shadow. It would be interesting to use the information regarding shadow points from the method presented here to determine if shadows are present and, if so, in which direction should they be expected.

It would also be interesting to explore the possibilities of using more than color information as input to the algorithm, for example, thermal information from an IR-camera or optical flow.



# Bibliography

- [1] Chandrika and Kamath. Robust techniques for background subtraction in urban traffic video. In *Proceedings of SPIE, Visual Communications and Image Processing 2004*, volume 5308, pages 881–892, January 2004.
- [2] Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, Department of Electrical Engineering and Computer Science U.C. Berkeley, April 1998.
- [3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, 1995. ISBN 0-19-853864-2.
- [4] P. Blauensteiner, H. Wildenauer, A. Hanbury, and M. Kampel. On colour spaces for change detection and shadow suppression. In *Proc. of the 11th CVWWS6: Computer Winter Vision Workshop*, pages 117–123, February 6-8, 2006.
- [5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
- [6] Allan Hanbury and Jean Serra. A 3d-polar coordinate colour representation suitable for image analysis. Technical Report PRIP-TR-77, Vienna University of Technology, 2002.
- [7] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proceedings of IEEE ICCV'99 FRAME-RATE Workshop*, 1999.
- [8] Björn Johansson. *Low Level Operations and Learning in Computer Vision*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, December 2004. Dissertation No. 912, ISBN 91-85295-93-0.
- [9] Erum A. Khan and Erik Reinhard. A survey of color spaces for shadow identification. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 160–160, New York, NY, USA, 2004. ACM Press.
- [10] P. Kumar, K. Sengupta, and A. Lee. A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 100–105, 2002.
- [11] Nicolas Martel-Brisson and Andre Zaccarin. Moving cast shadow detection from a gaussian mixture shadow model. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 643–648, Washington, DC, USA, 2005. IEEE Computer Society.

- [12] N. McFarlane and C. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, 8(3):187–193, 1995.
- [13] Wayne P. Power and Johann A. Schoonees. Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, 2002.
- [14] Charles Poynton. Frequently asked questions about color, 1999. <http://www.poynton.com/ColorFAQ.html>.
- [15] Charles Poynton. *Digital Video and HDTV Algorithms and Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [16] Andrea Prati, Ivana Mikic, Mohan M. Trivedi, and Rita Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(7):918–923, 2003.
- [17] Alvy Ray Smith. Color gamut transform pairs. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 12–19, New York, NY, USA, 1978. ACM Press.
- [18] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 246–252. IEEE, 1999.
- [19] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & sons, second edition, 1982.

## Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

## Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>